

# Finding Multiple Optima of Particle Accelerator Simulations

Jeffrey Larson Stefan Wild

Argonne National Laboratory

June 14, 2015

## Motivation

- ▶ We want to identify distinct, “high-quality”, local minimizers of the problem

$$\text{minimize } f(x)$$

$$l \leq x \leq u$$

$$x \in \mathbb{R}^n$$

- ▶ High-quality can be measured by more than the objective.
- ▶ The simulation  $f$  is likely using parallel resources, but it does not scale to the entire machine.
- ▶ Derivatives of  $f$  may or may not be available.



# Content

- I Motivation & review of other methods
- II Our method's description & theory
- III Performance metrics & numerical results



# Content

- I Motivation & review of other methods
- II Our method's description & **theory**
- III Performance metrics & numerical results

We essentially want a global method.



# Global optimization is tough

Theorem (Törn and Žilinskas, *Global Optimization*, 1989)

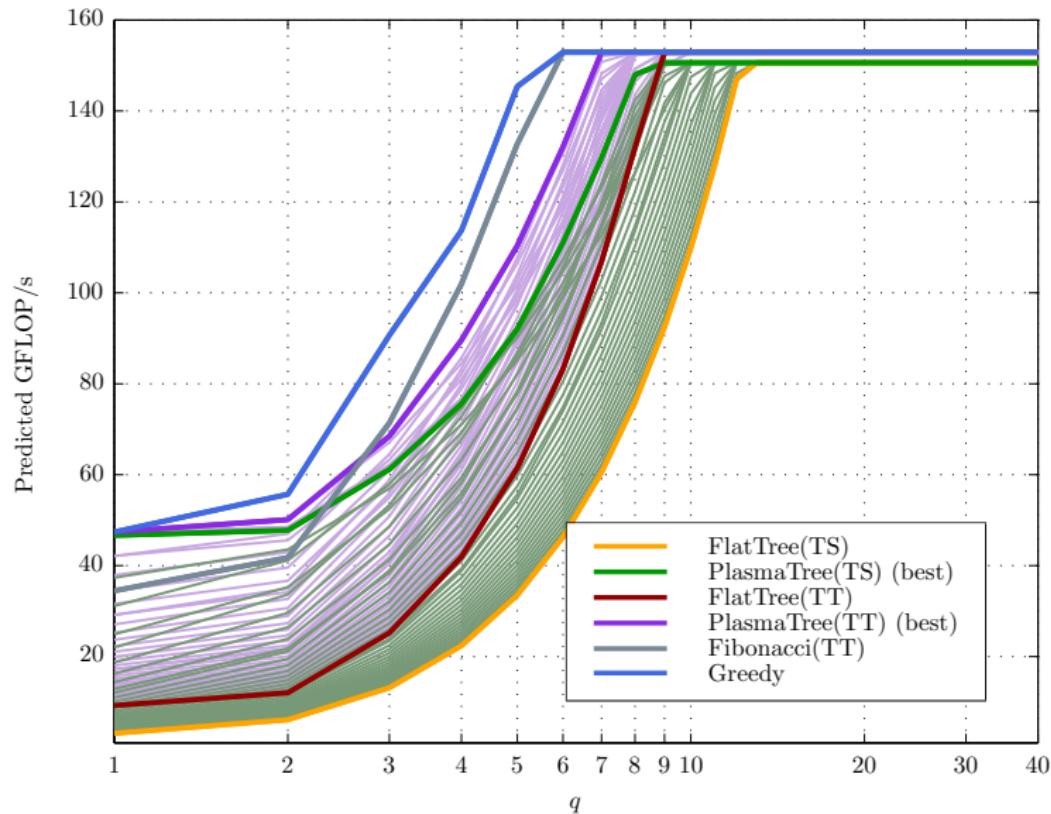
*An algorithm that converges to the global minimum of any continuous  $f$  if and only if the algorithm generates iterates that are dense in  $\mathcal{D}$ .*

Unless you can assume something about the function  $f$  or the domain  $\mathcal{D}$ , expect to wait a long time.

The theory can be more than just checking if the method generates iterates that are dense in the domain.



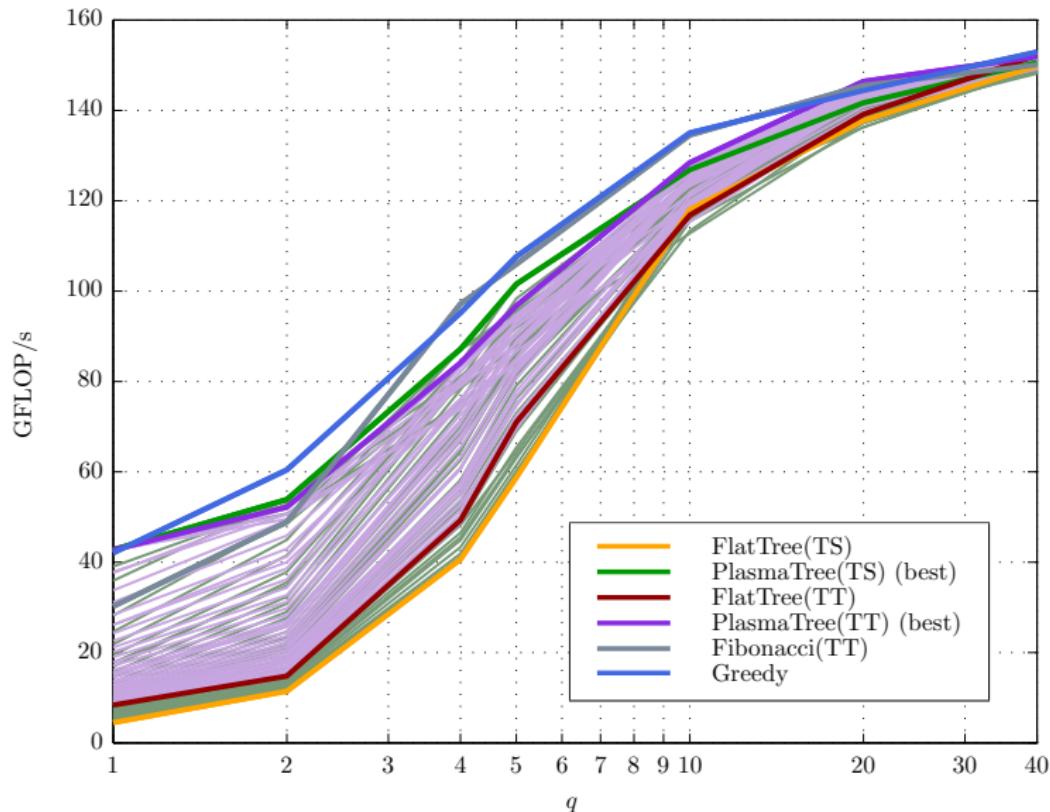
# Why concurrency? Tiled QR example



[Bouwmeester, et al., Tiled QR Factorization Algorithms, 2011]



# Why concurrency? Tiled QR example



[Bouwmeester, et al., Tiled QR Factorization Algorithms, 2011]



# Motivation



# Motivation

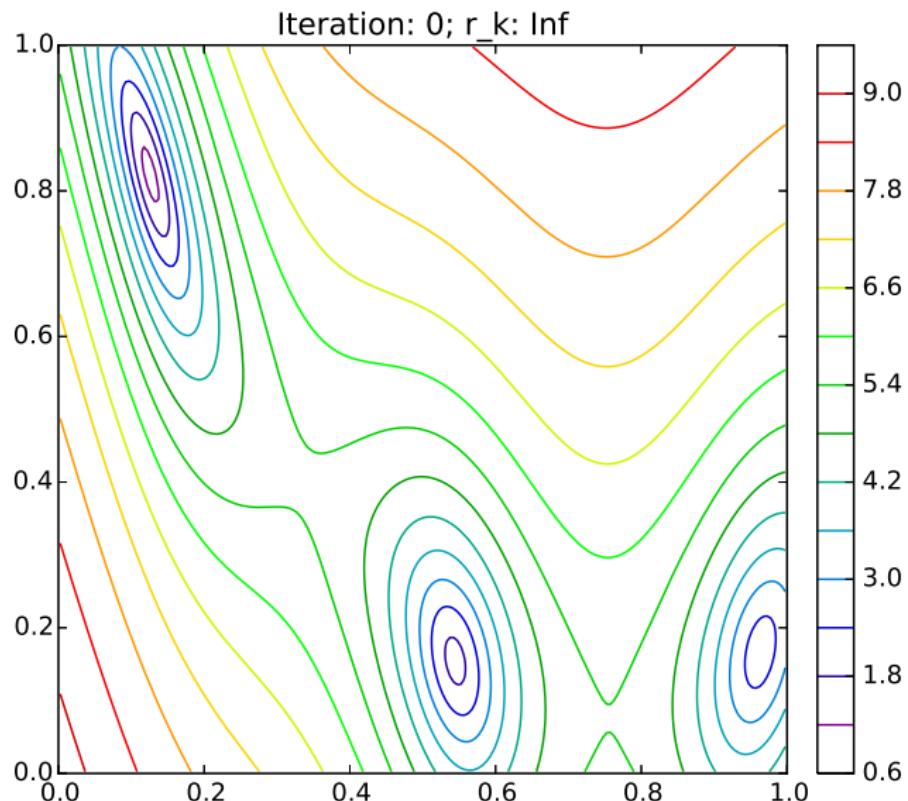


# Algorithms for global optimization

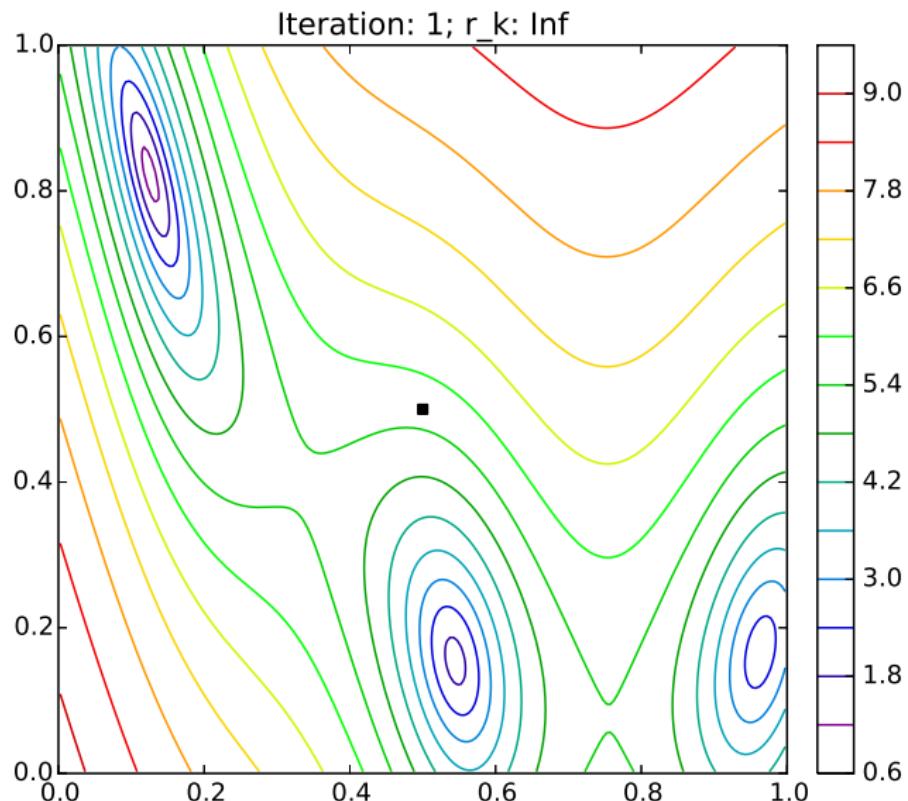
- ▶ Trade off between exploration and refinement
- ▶ Evolutionary Algorithms
- ▶ Direct Methods
- ▶ Multistart Methods



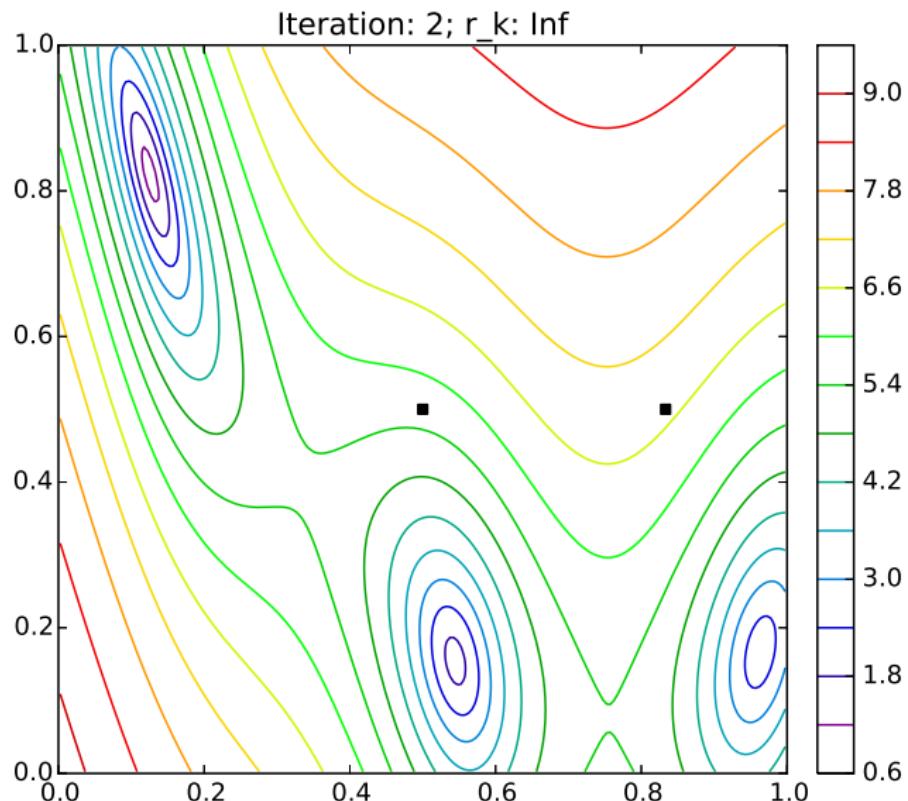
# Direct



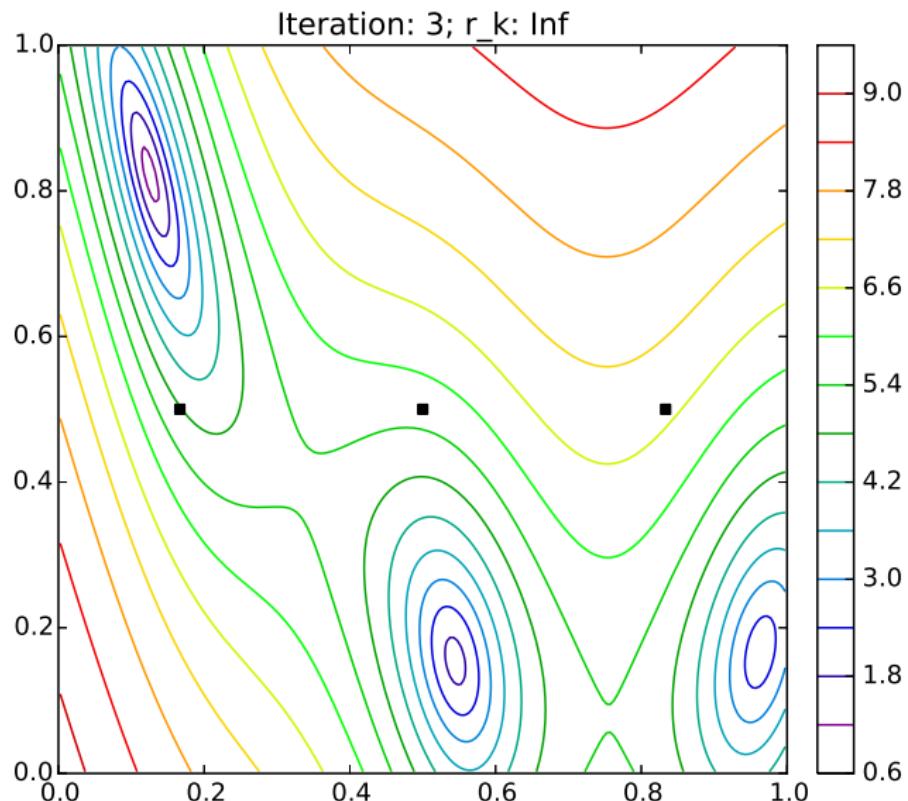
# Direct



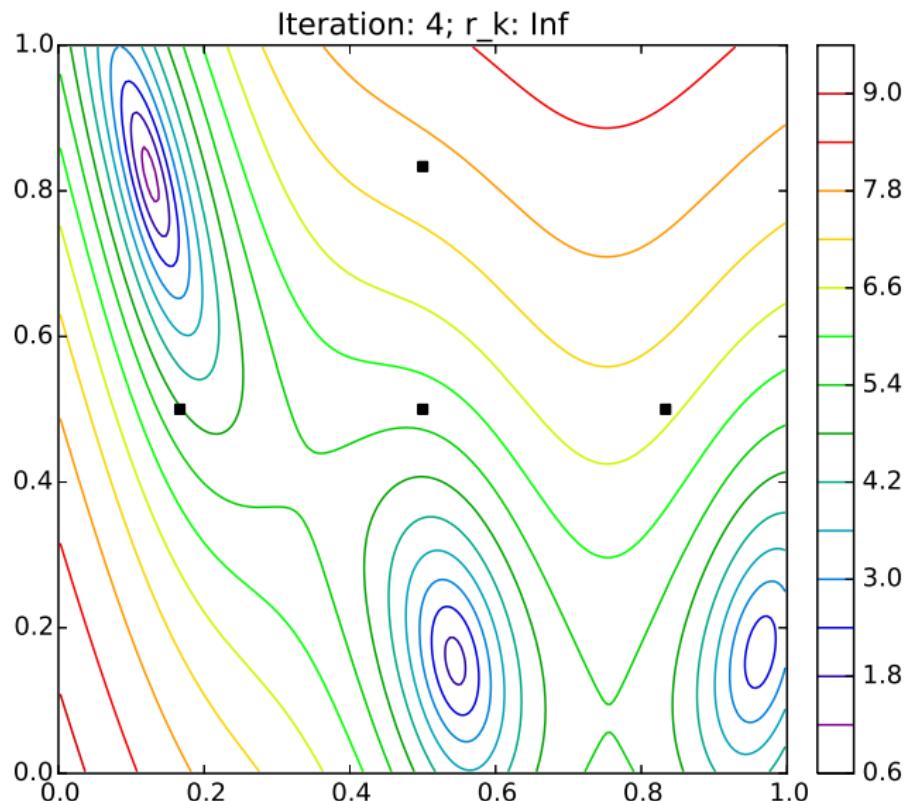
# Direct



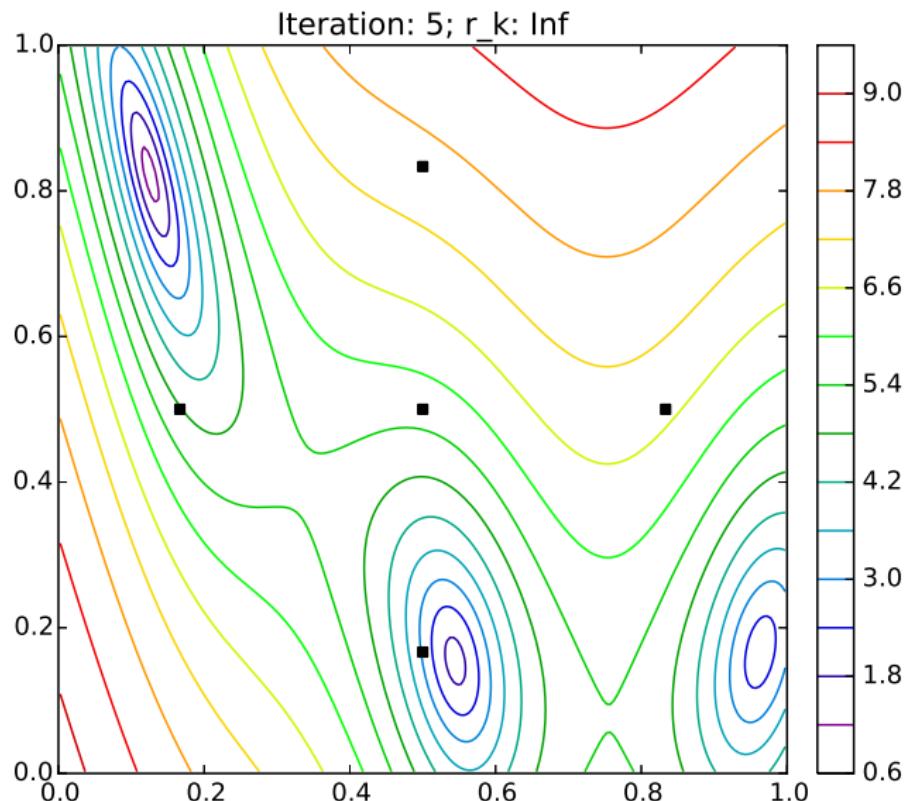
# Direct



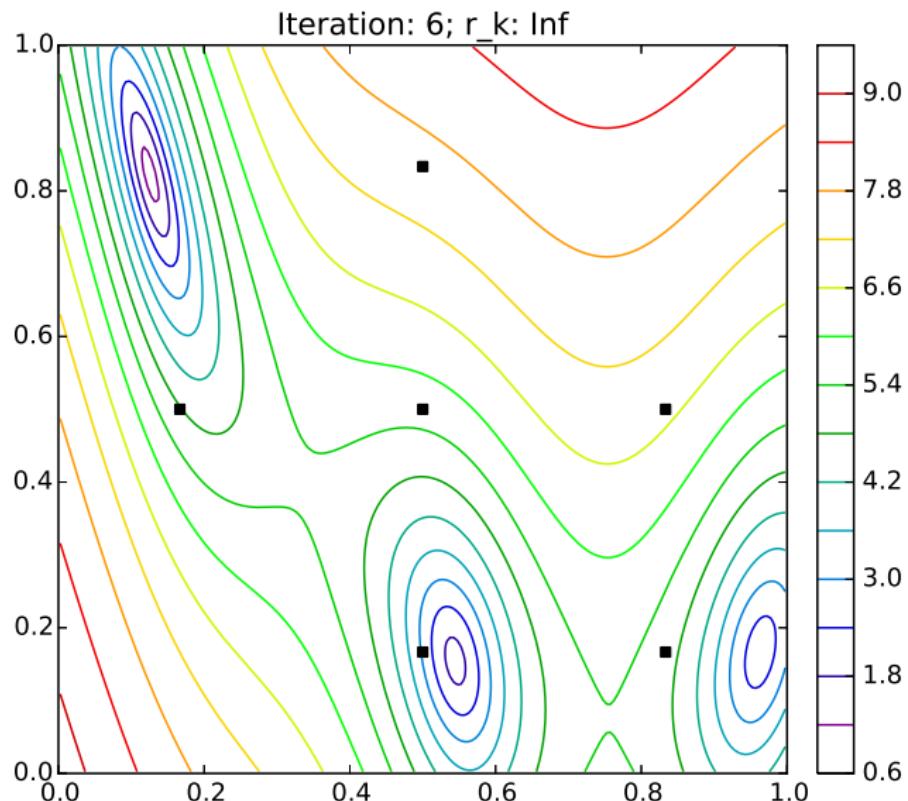
# Direct



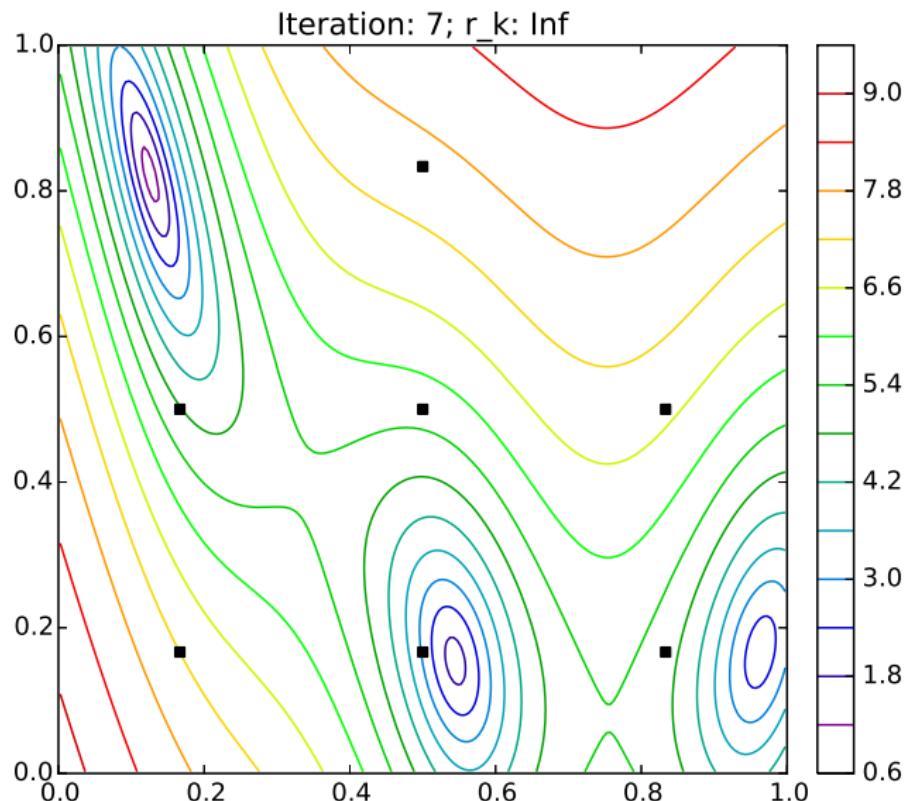
# Direct



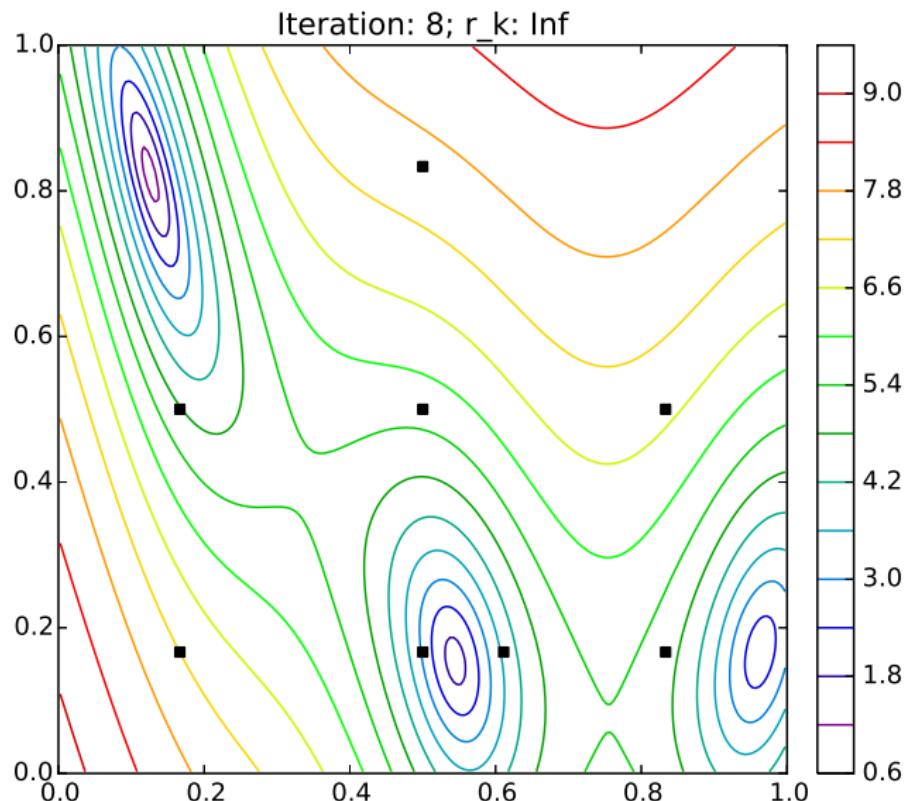
# Direct



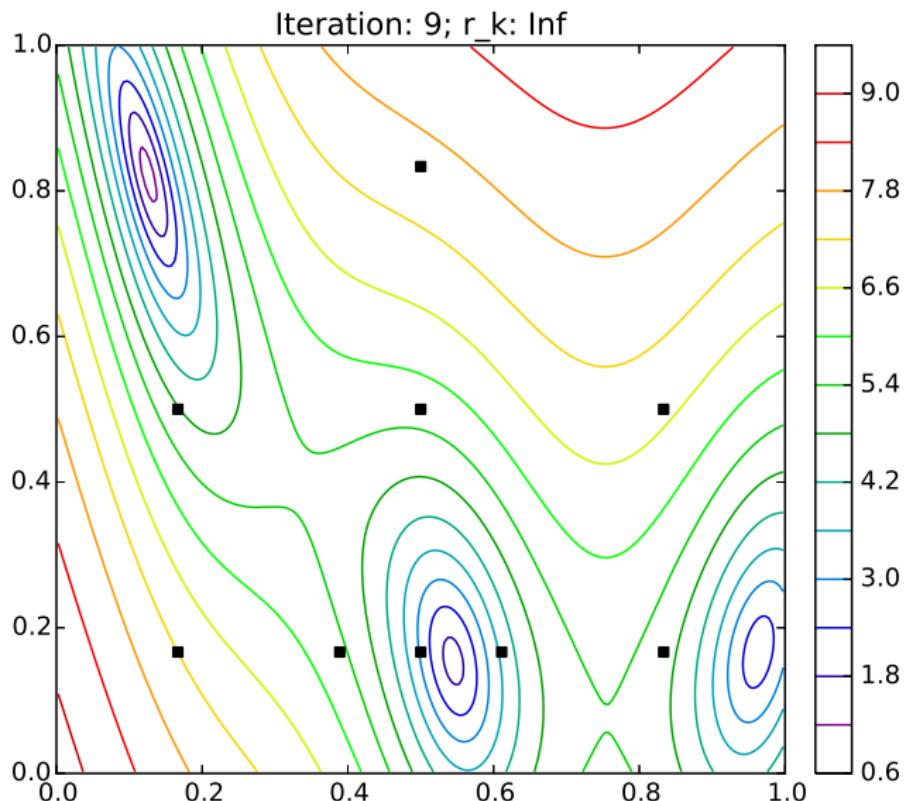
# Direct



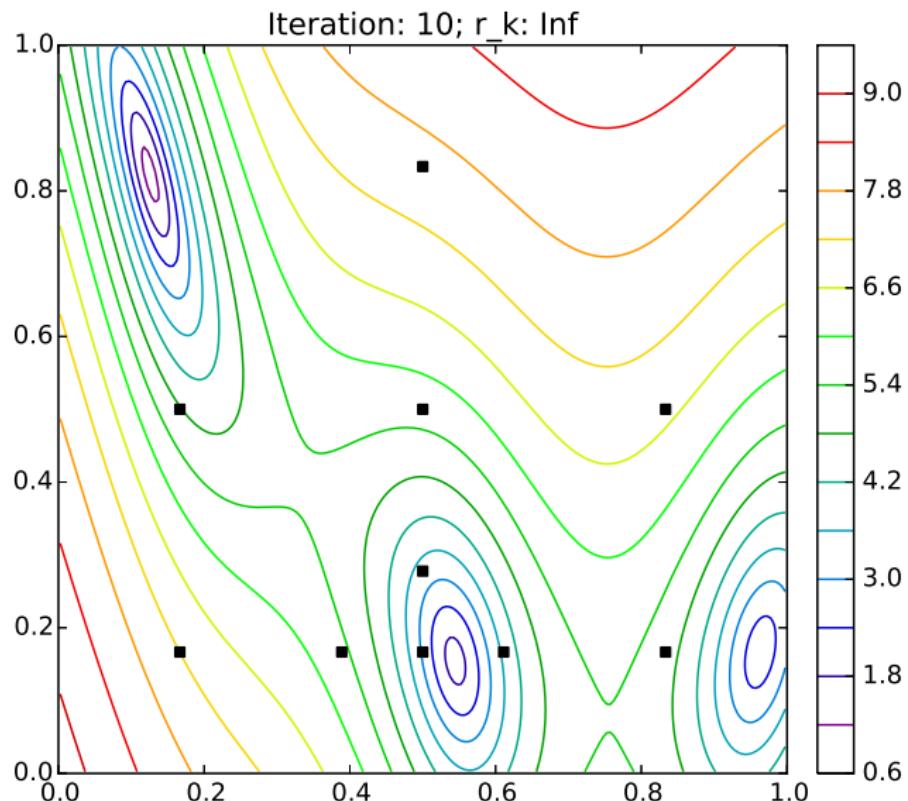
# Direct



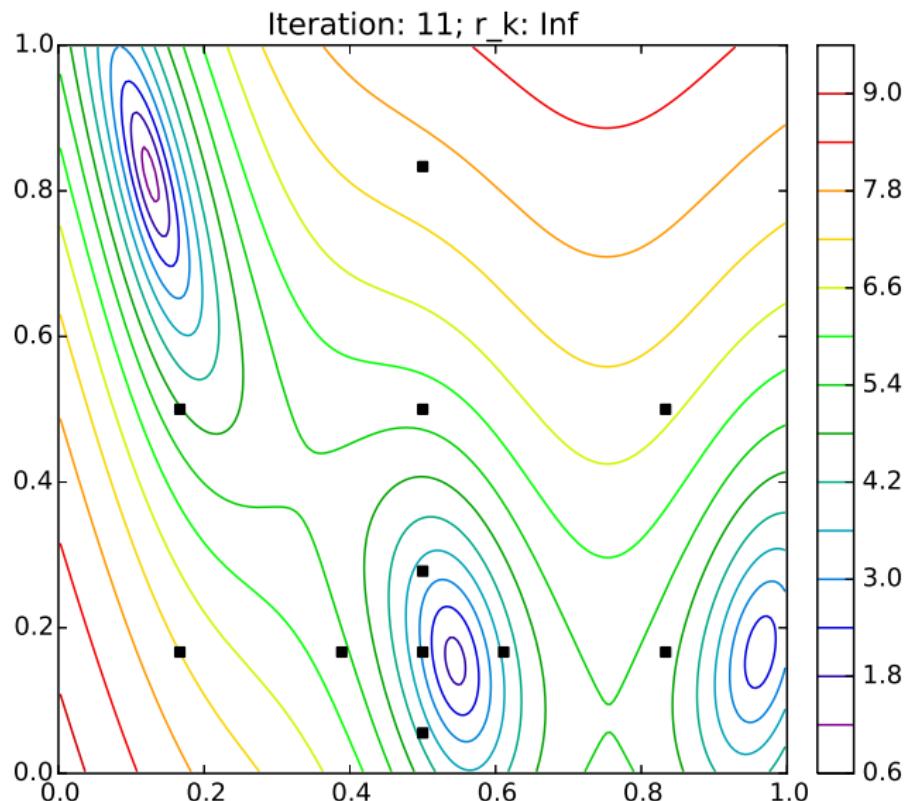
# Direct



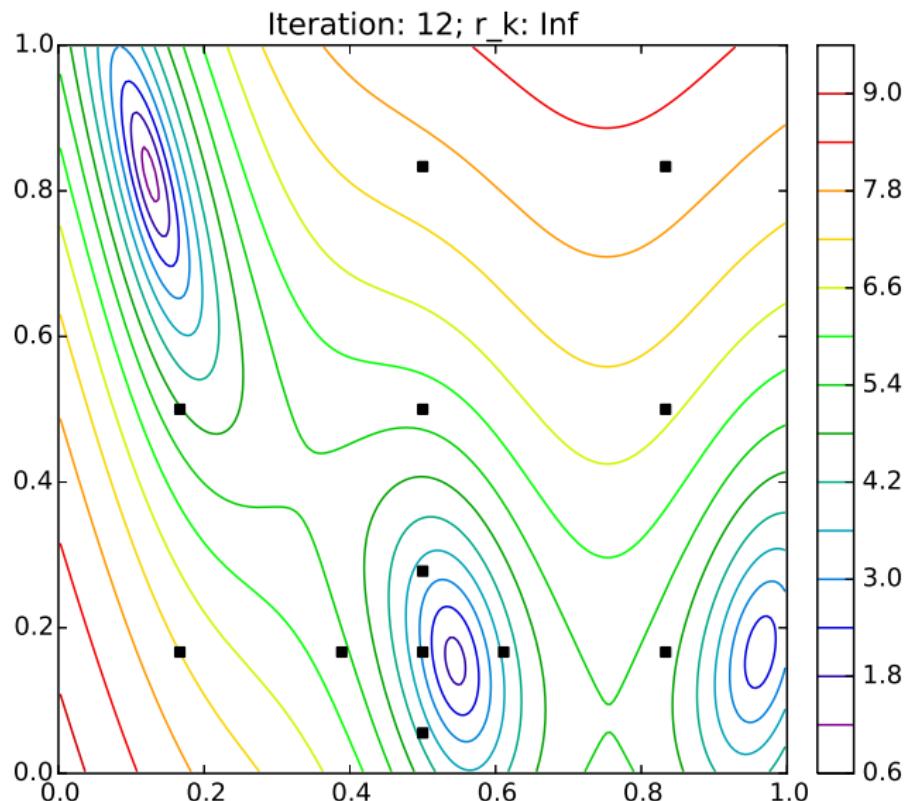
# Direct



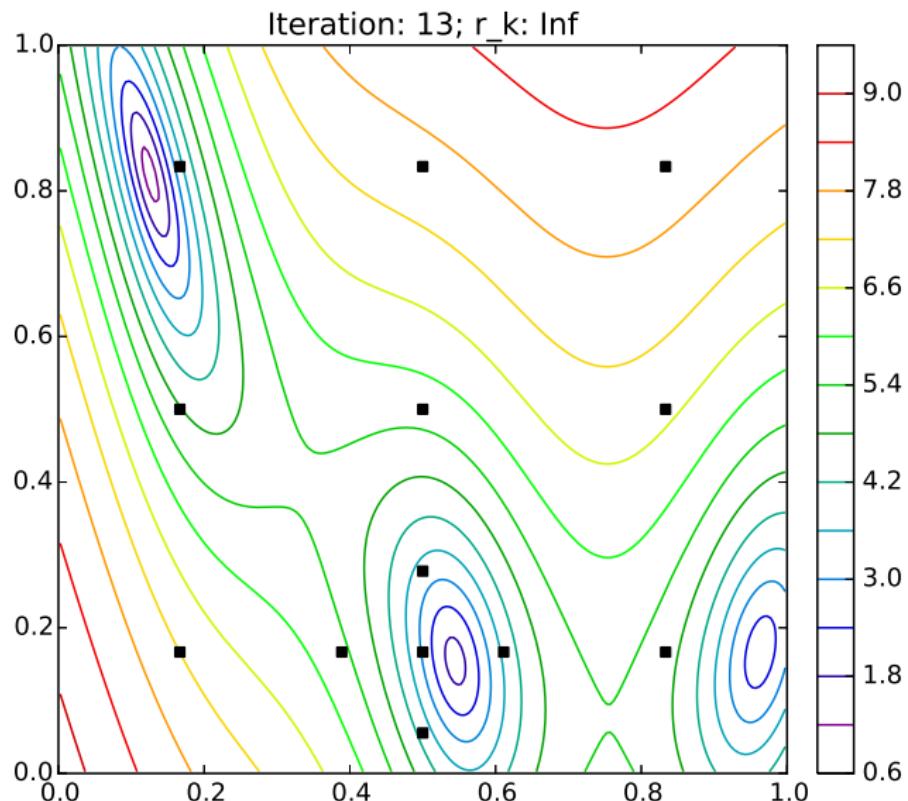
# Direct



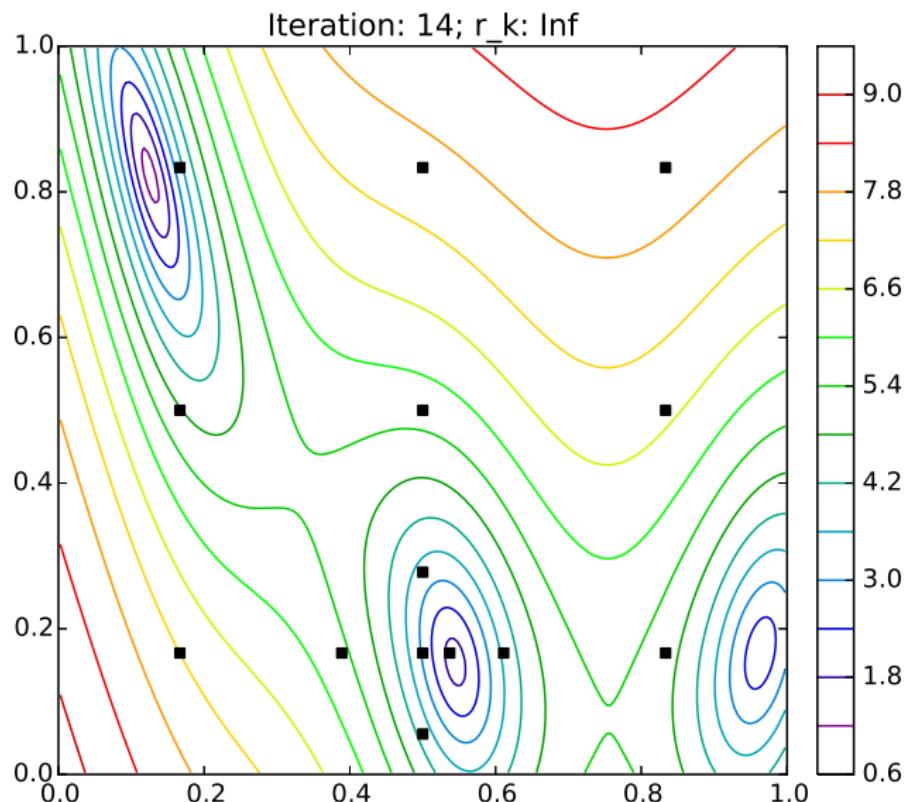
# Direct



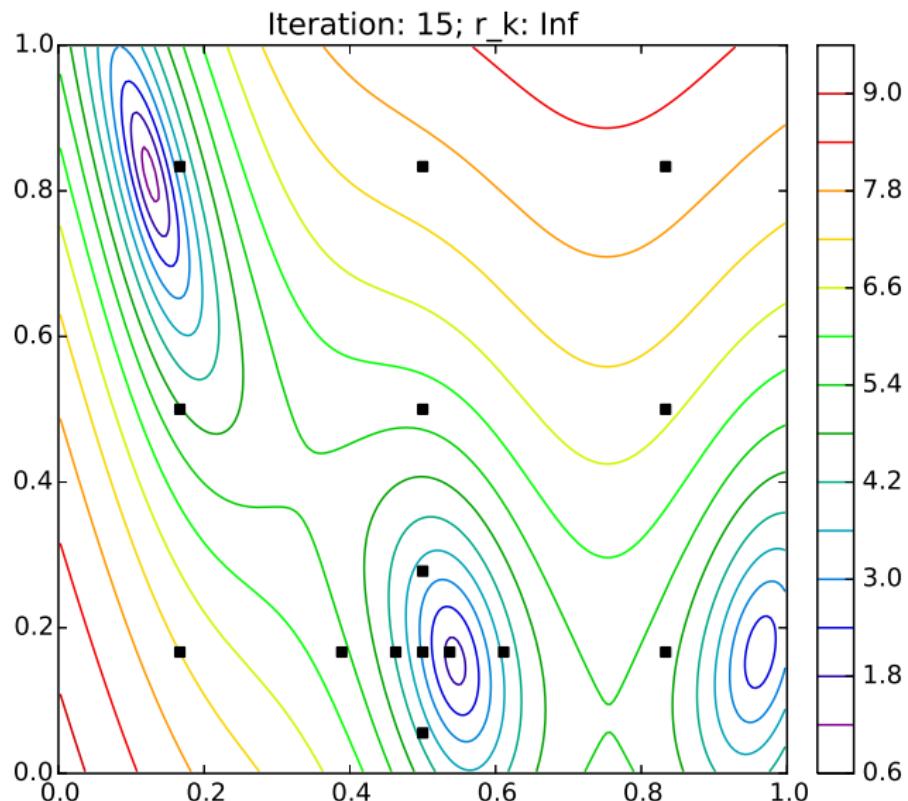
# Direct



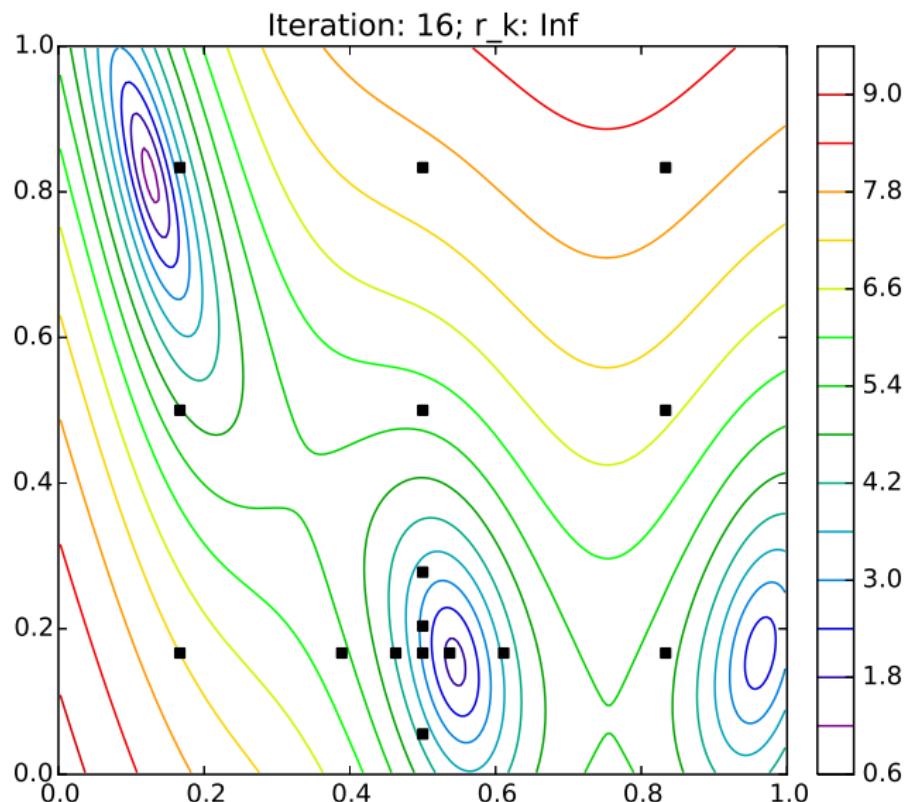
# Direct



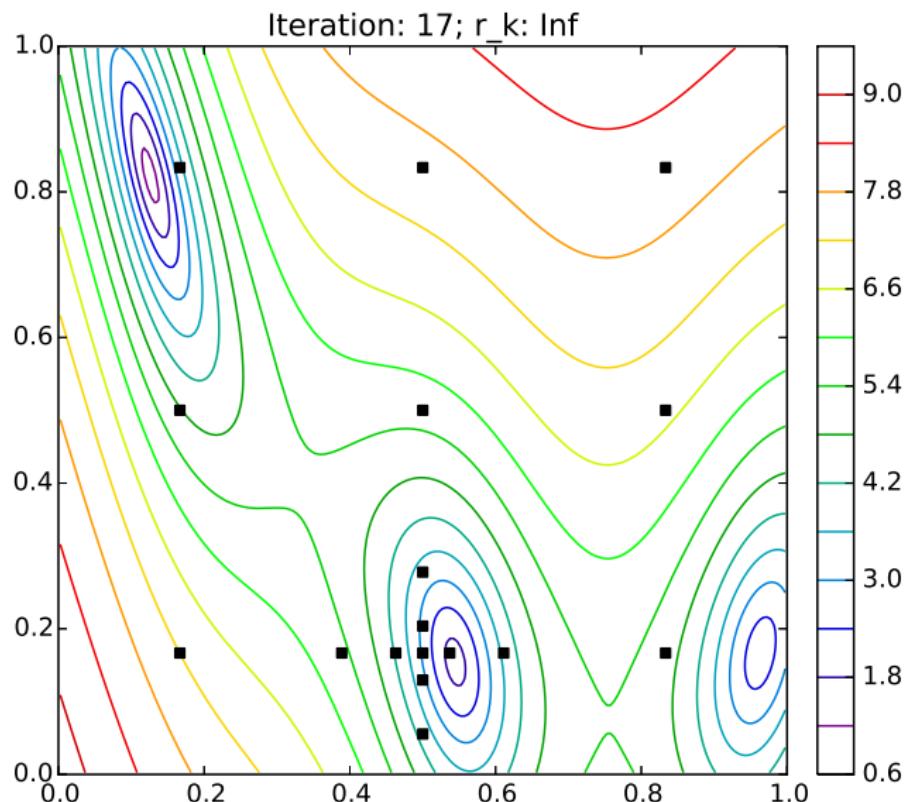
# Direct



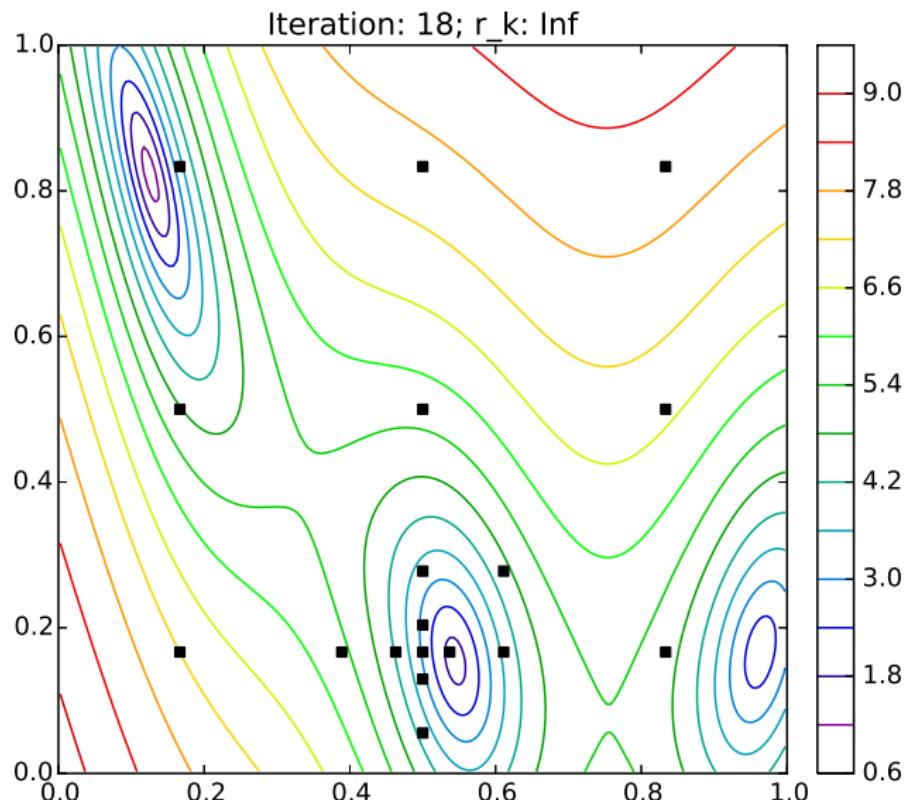
# Direct



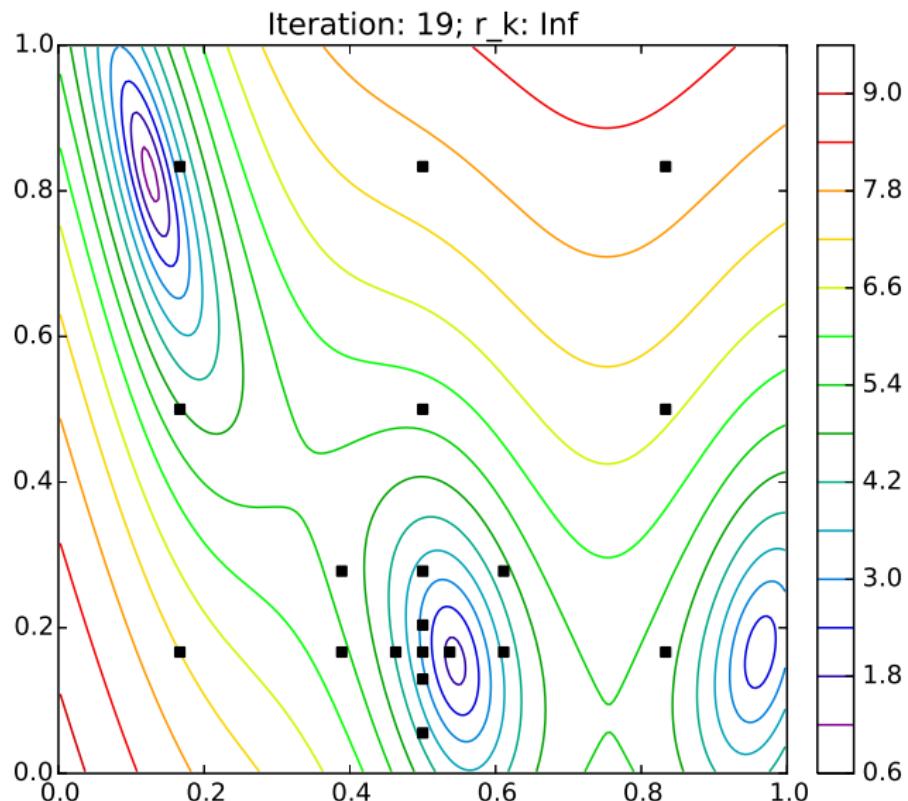
# Direct



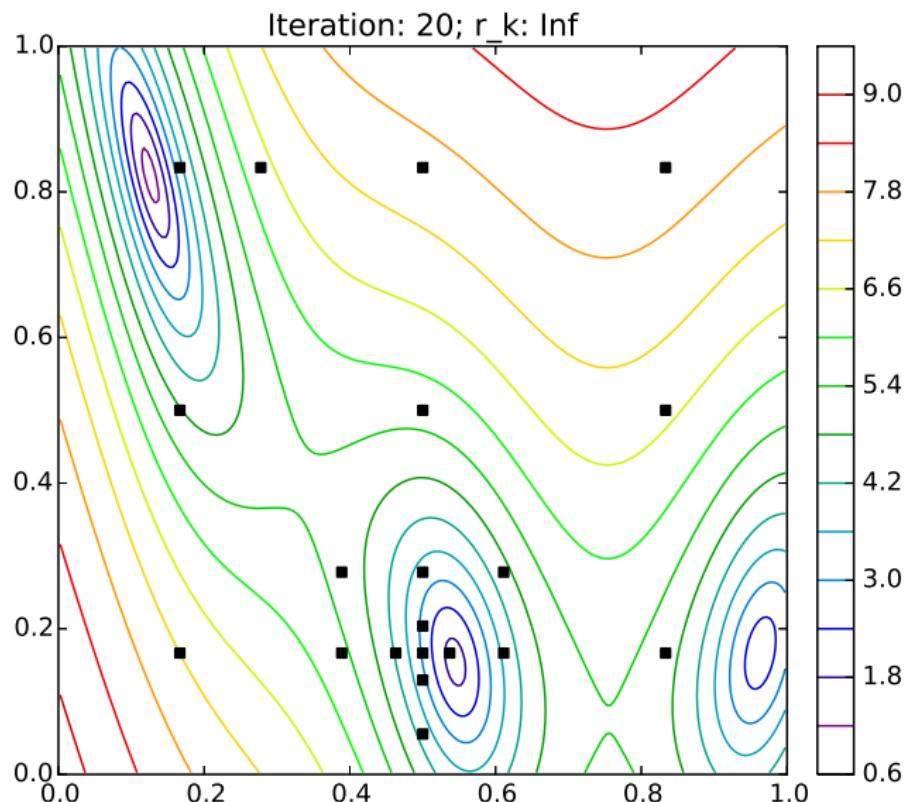
# Direct



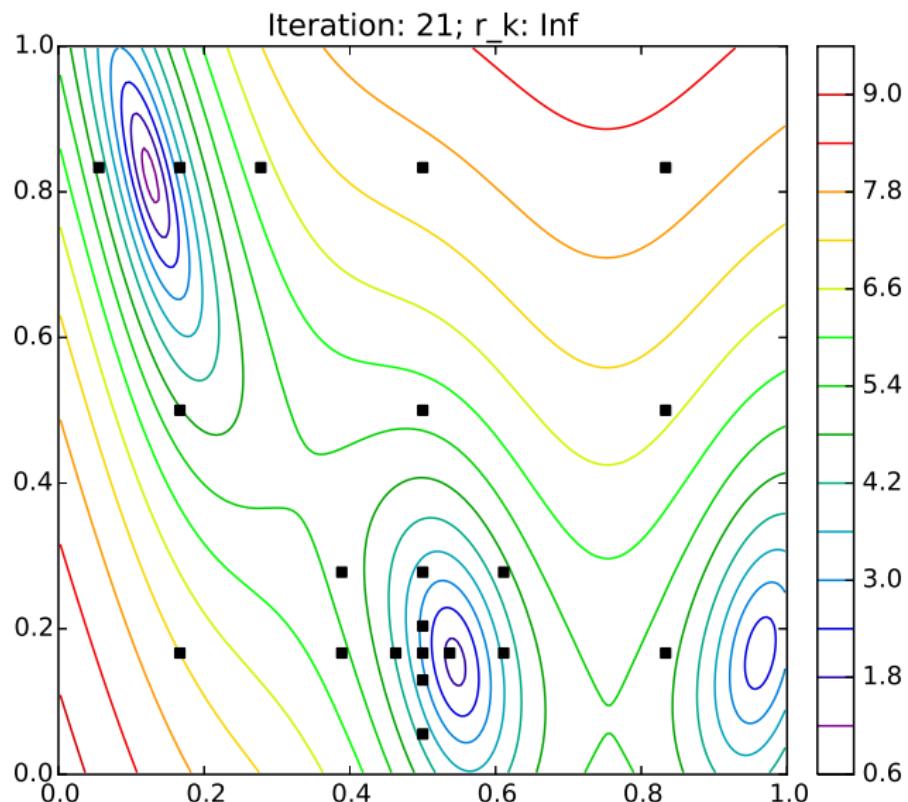
# Direct



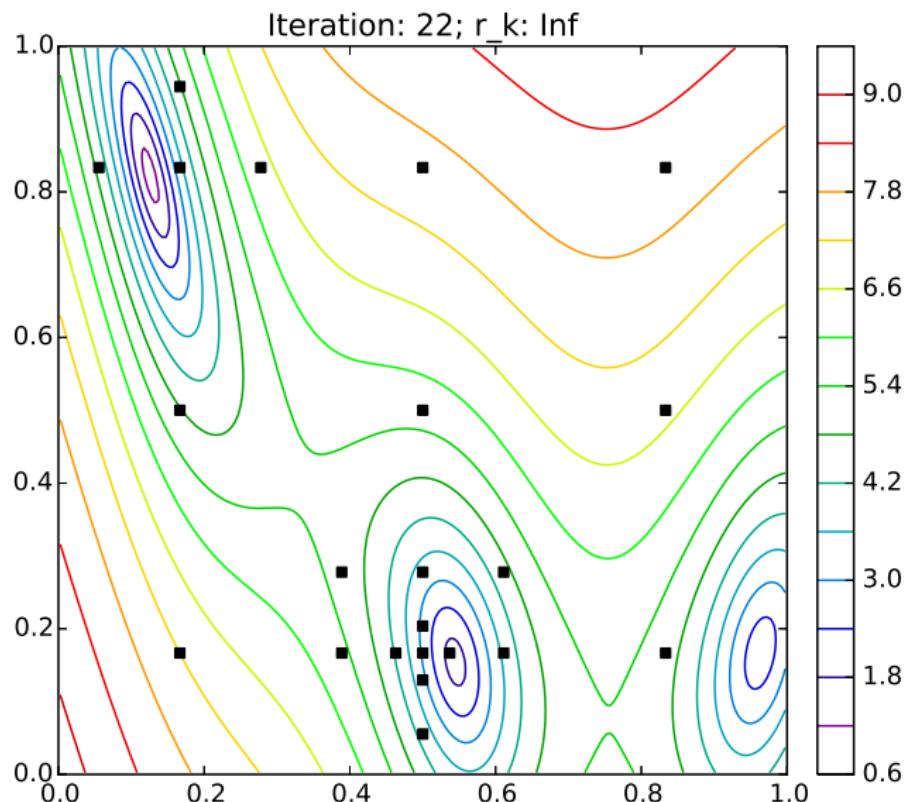
# Direct



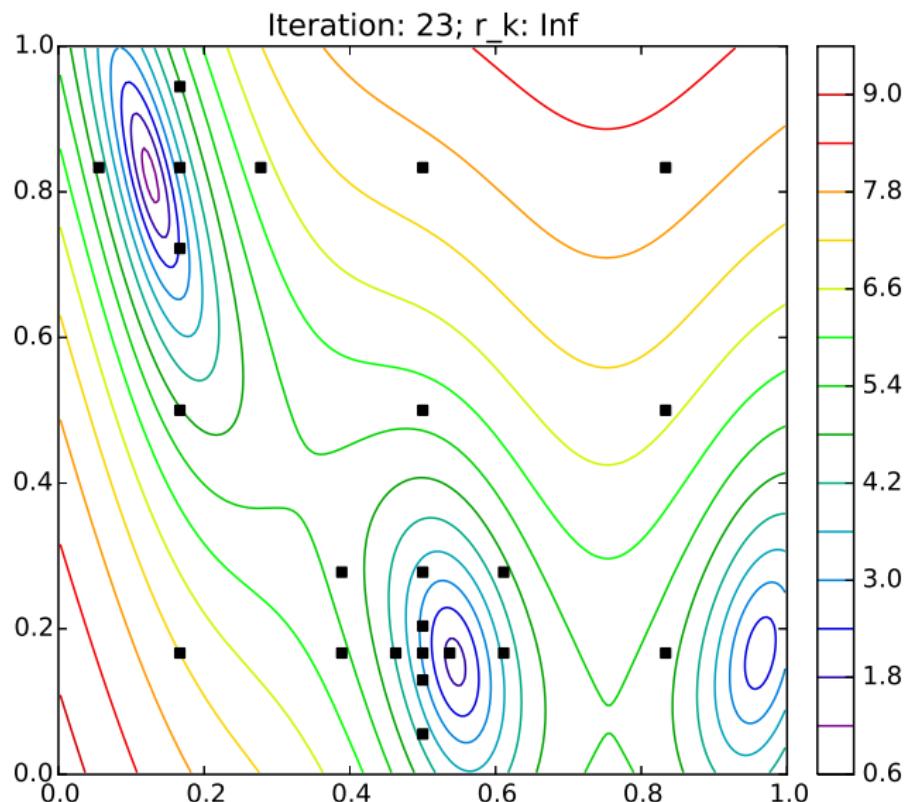
# Direct



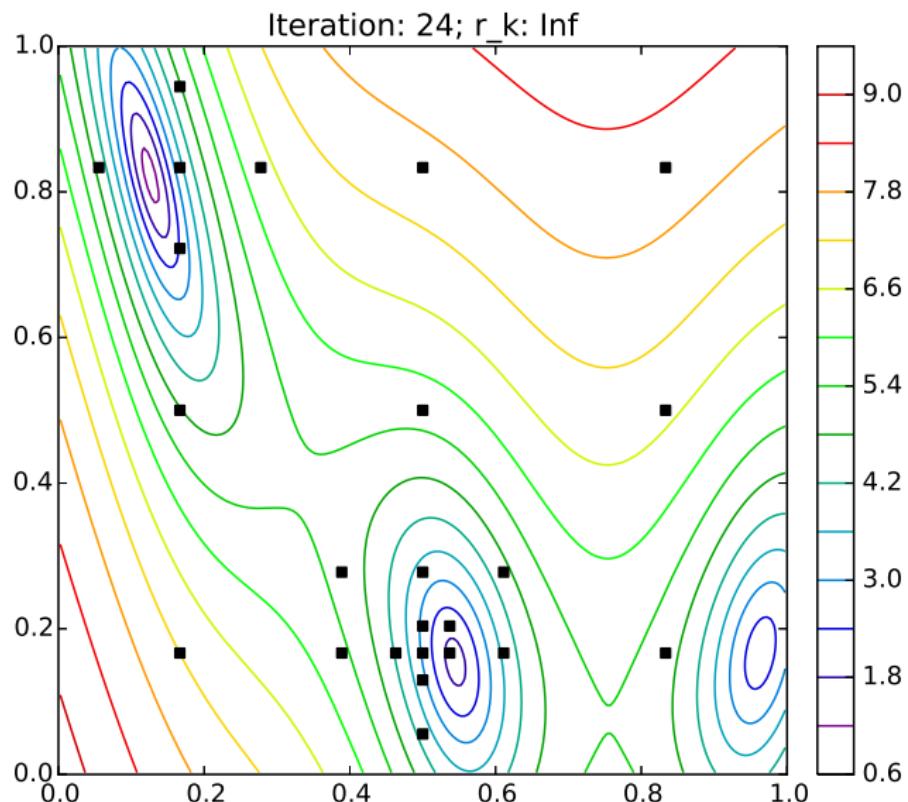
# Direct



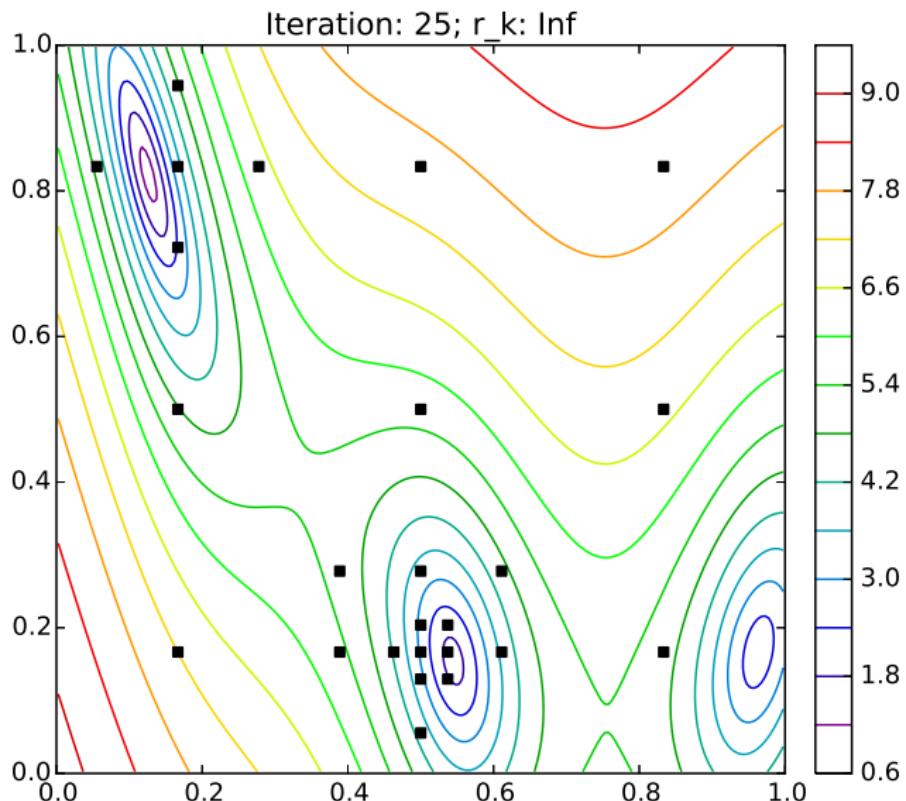
# Direct



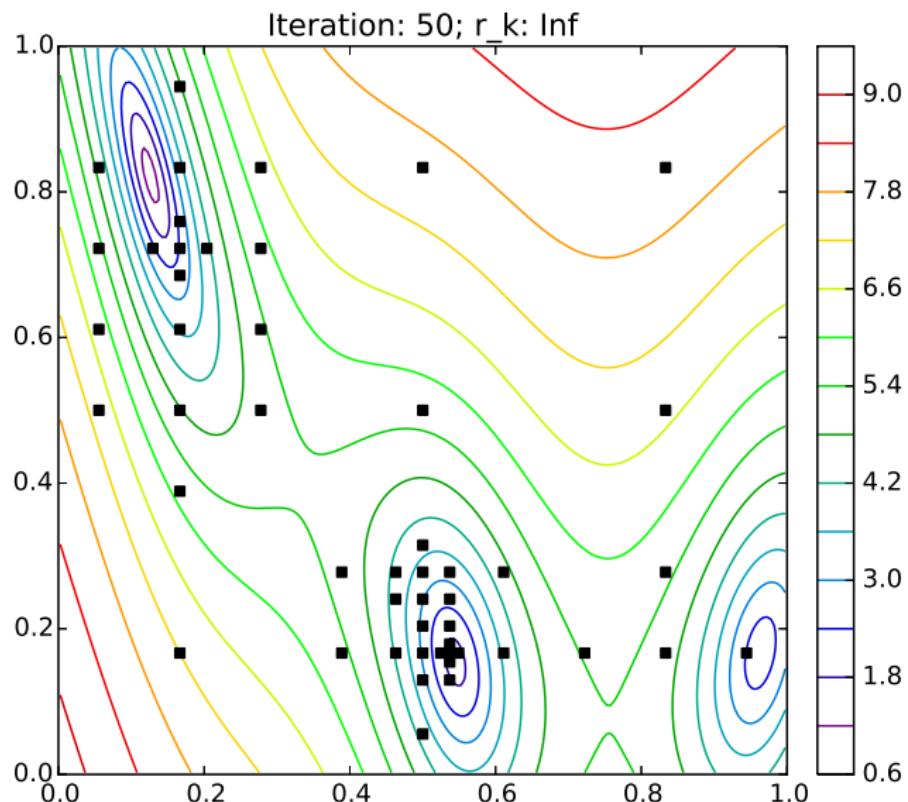
# Direct



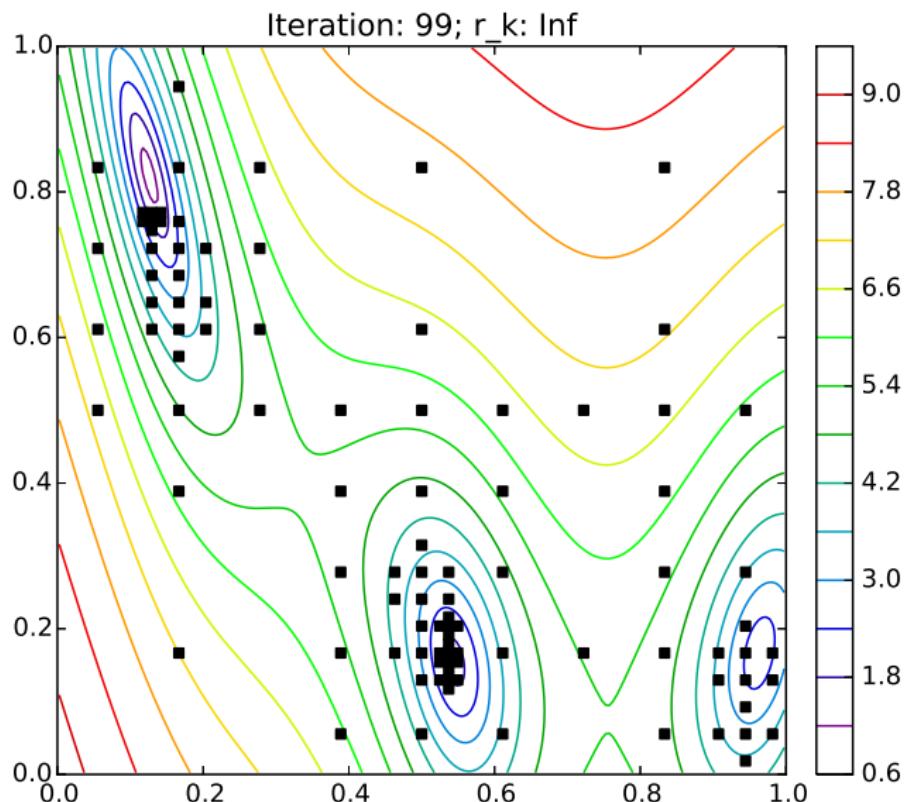
# Direct

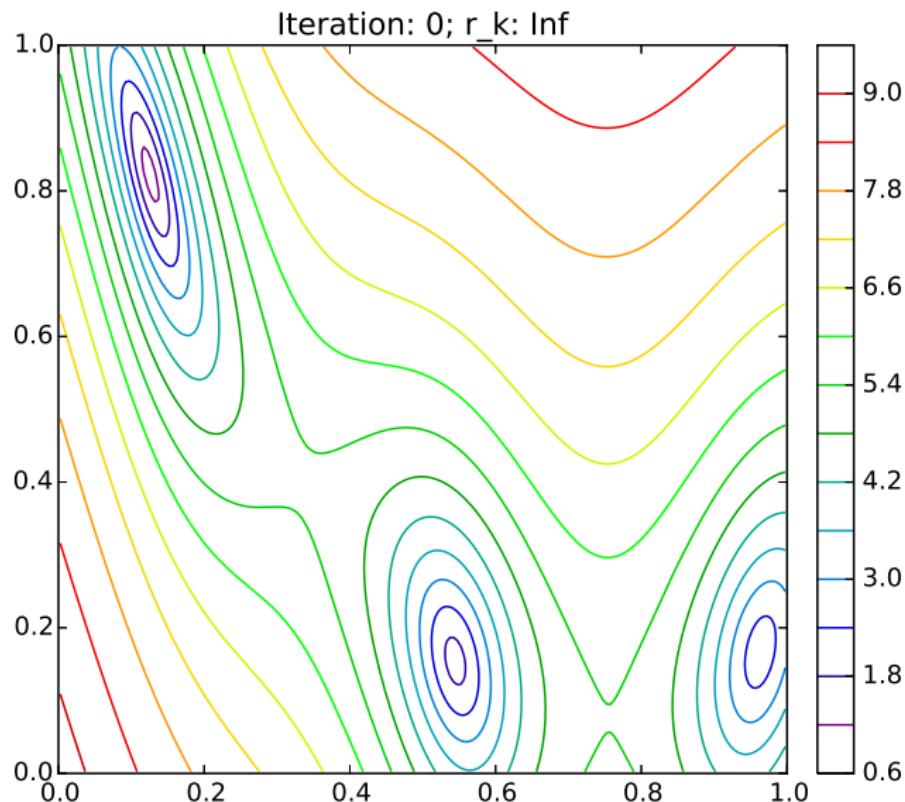


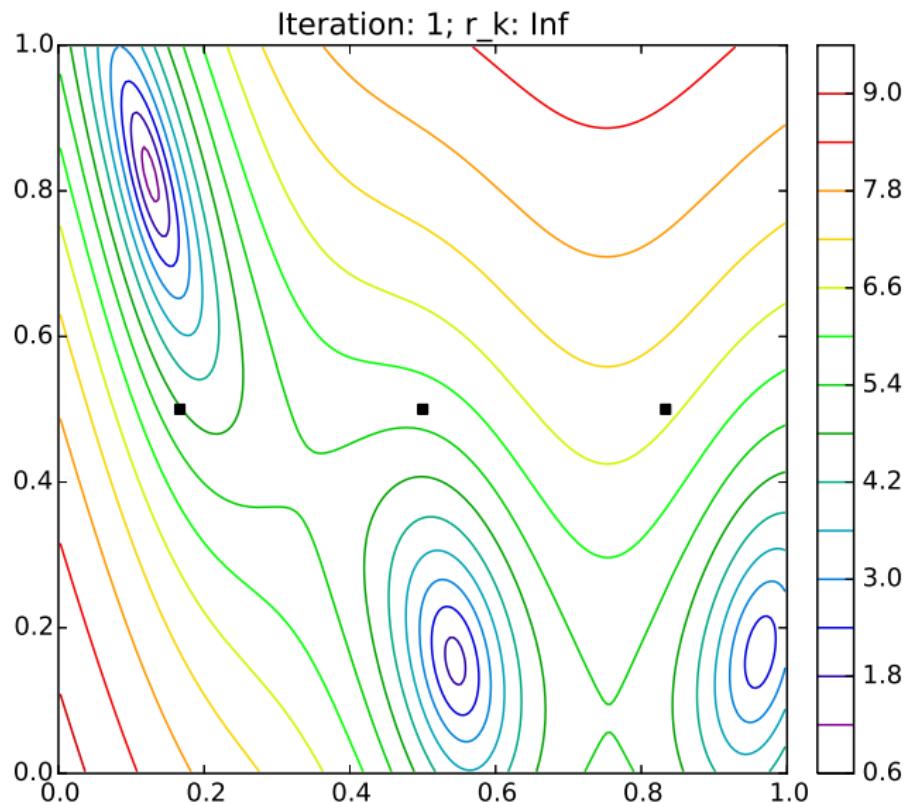
# Direct

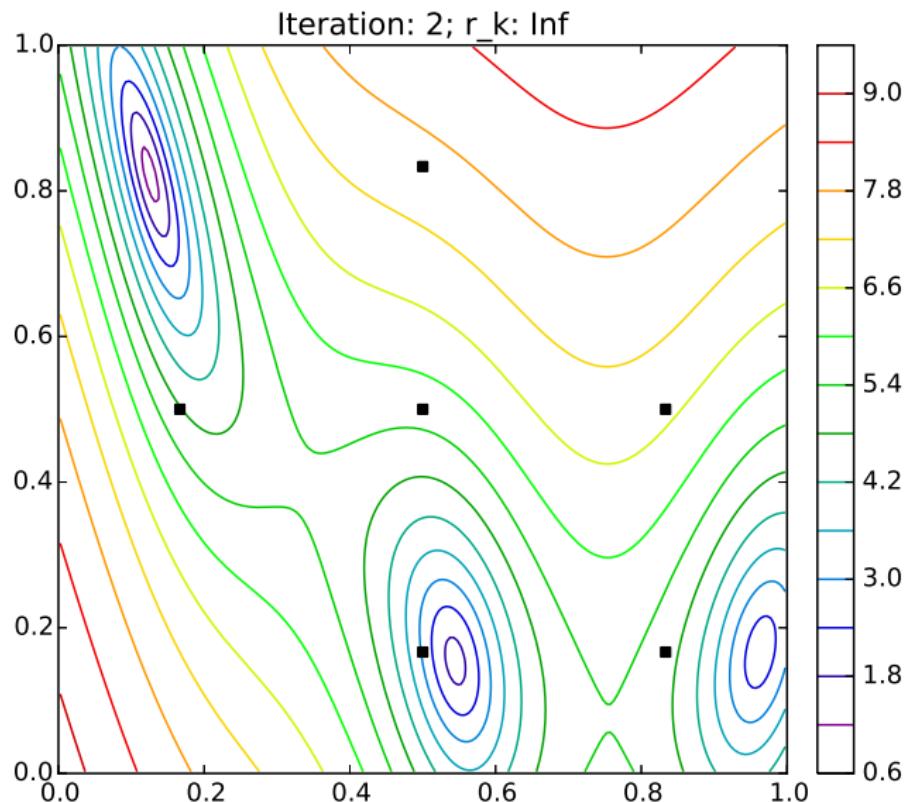


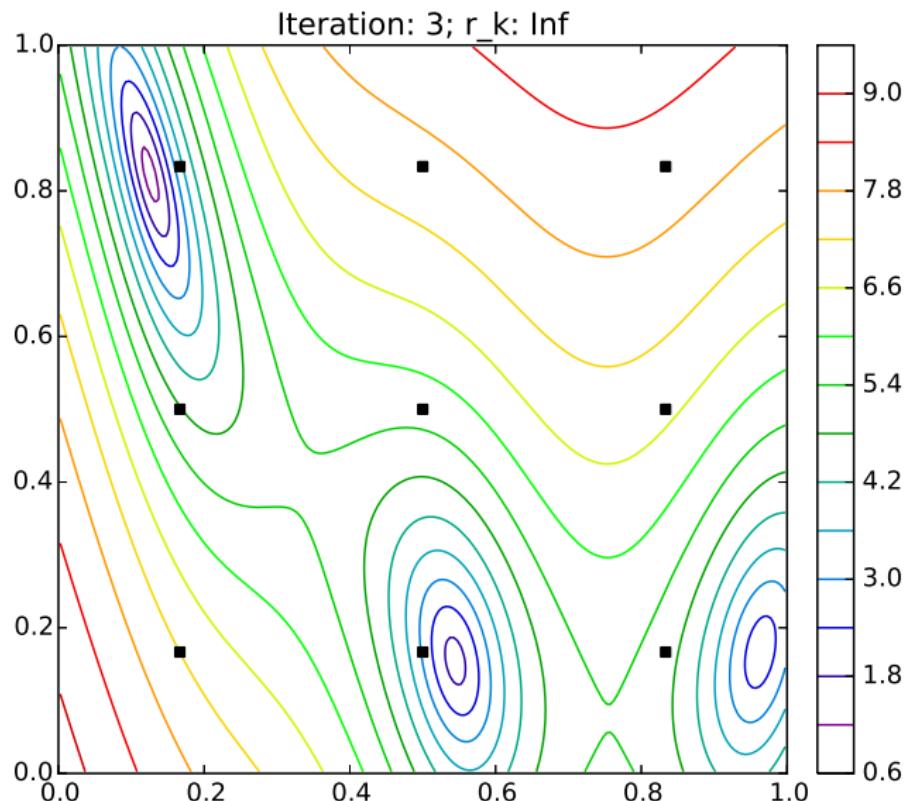
# Direct

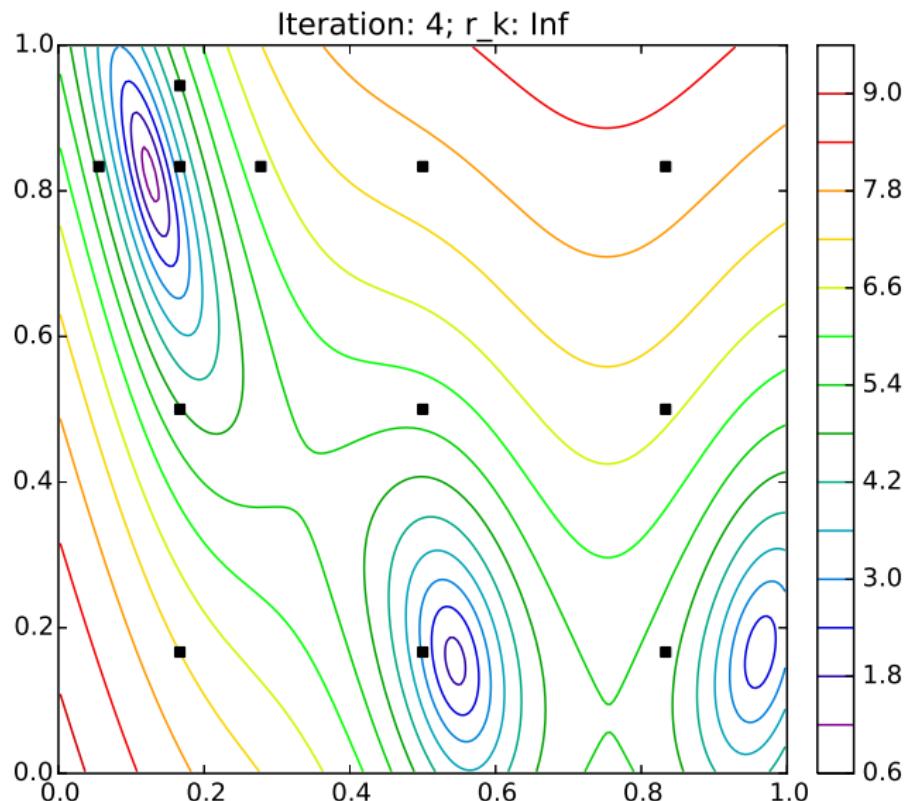


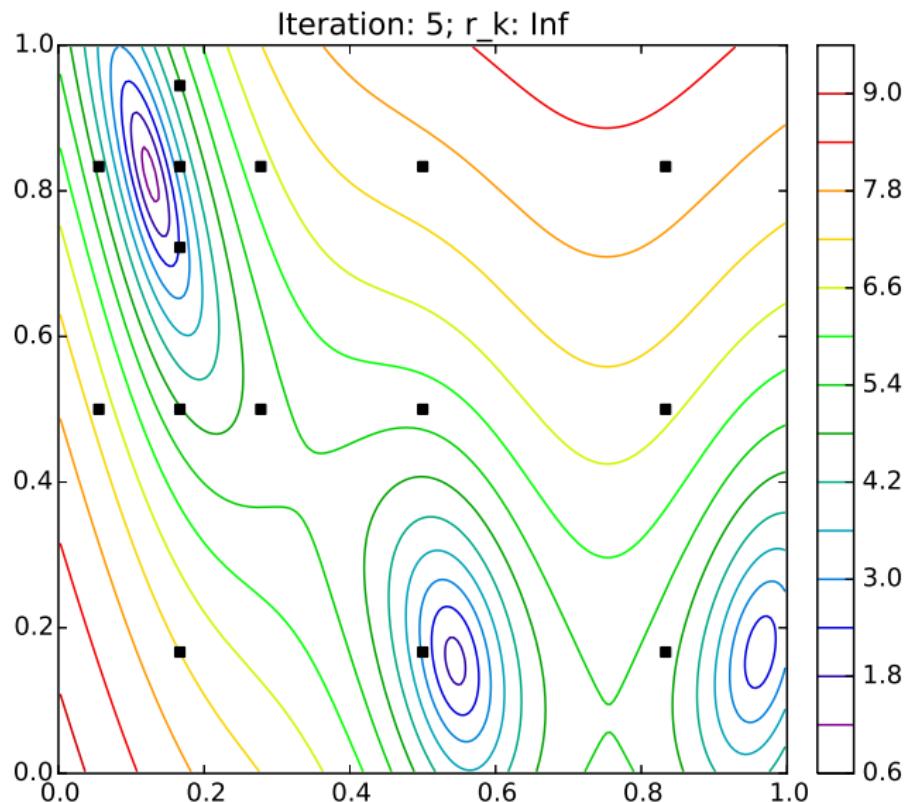


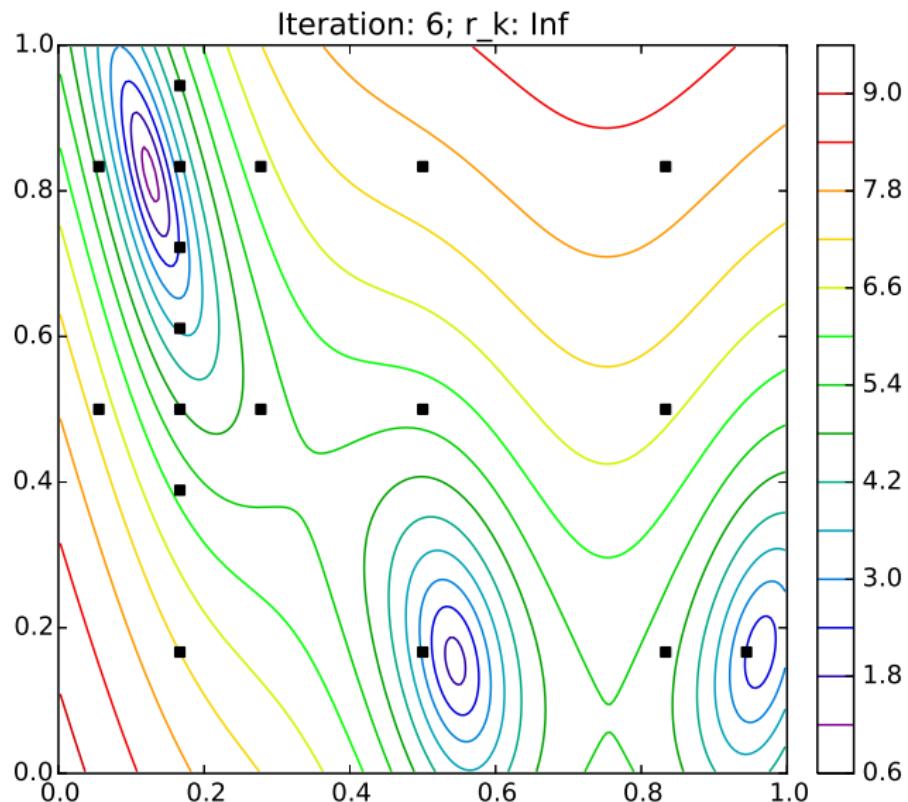


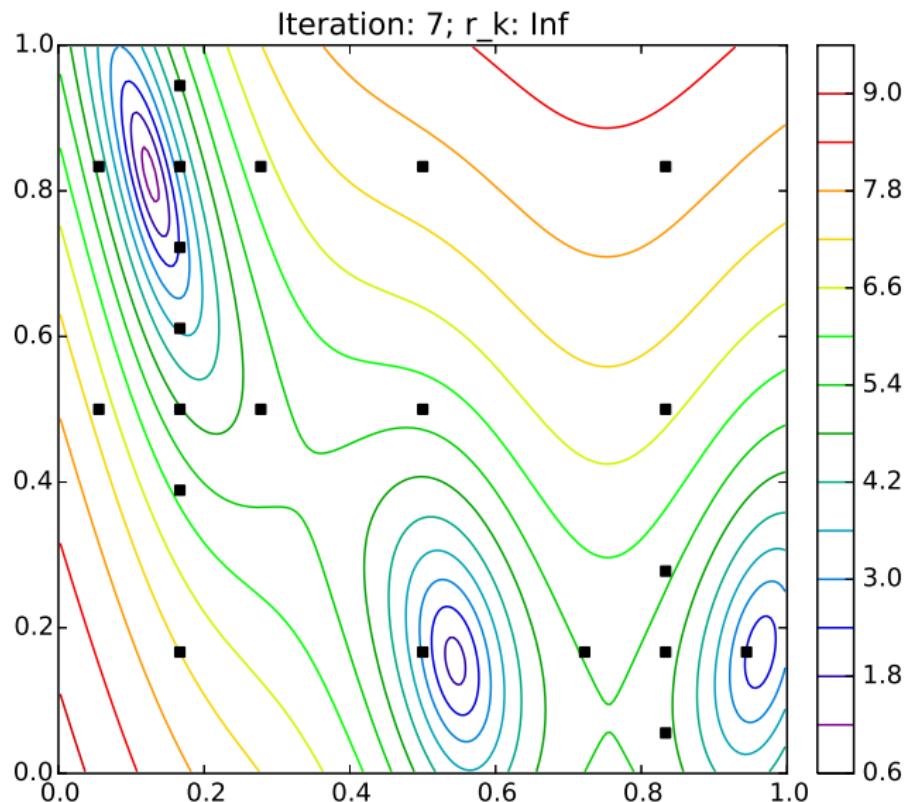


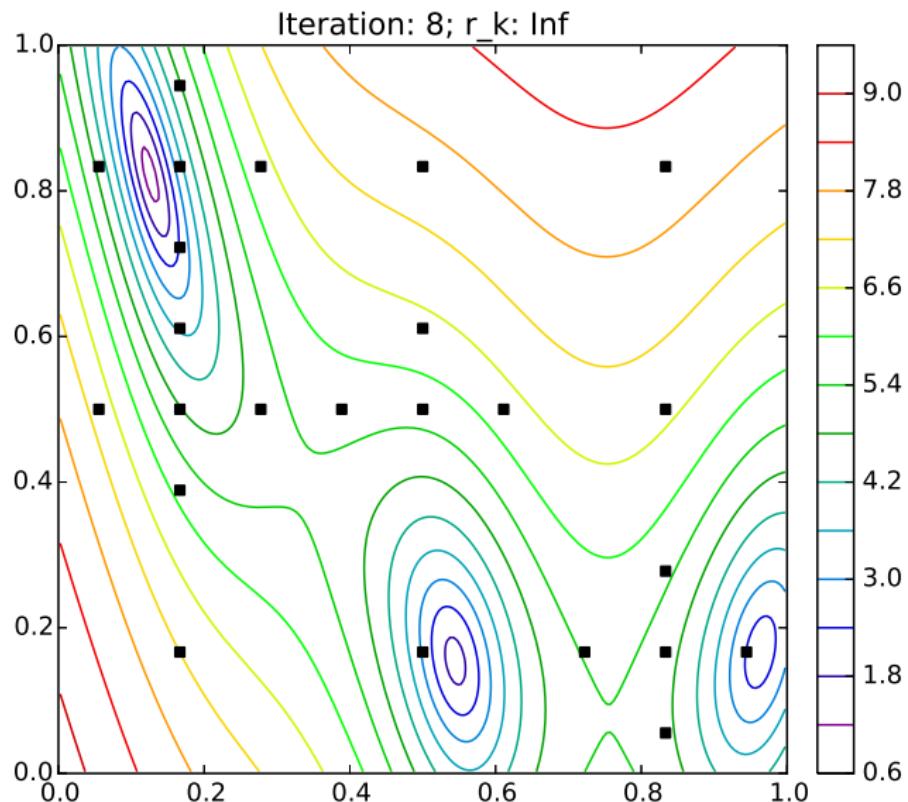


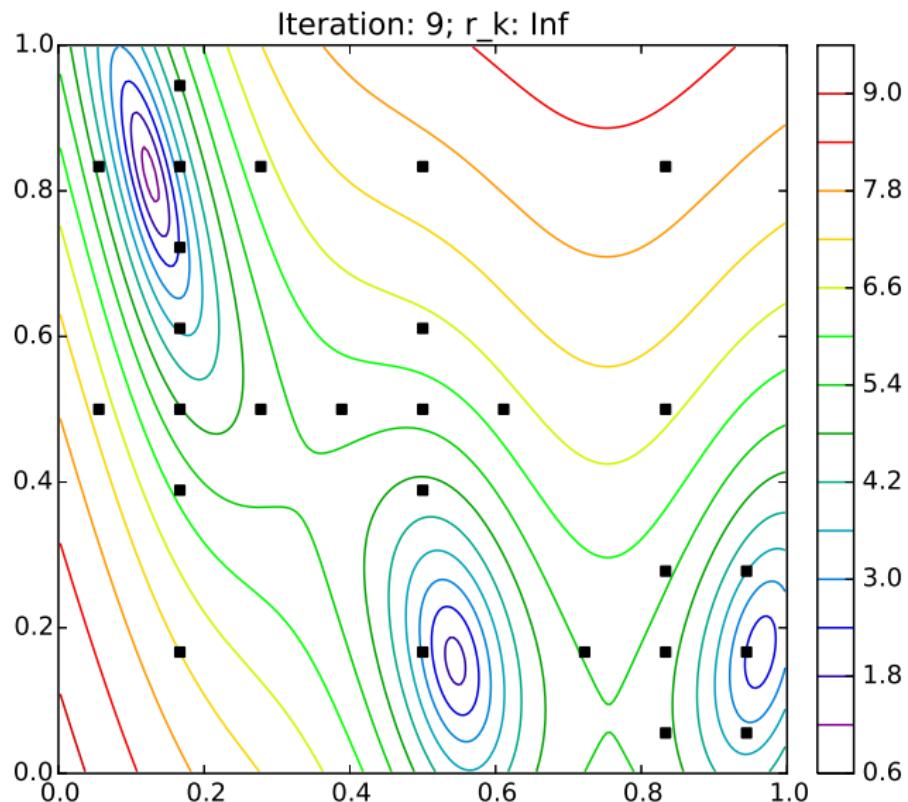


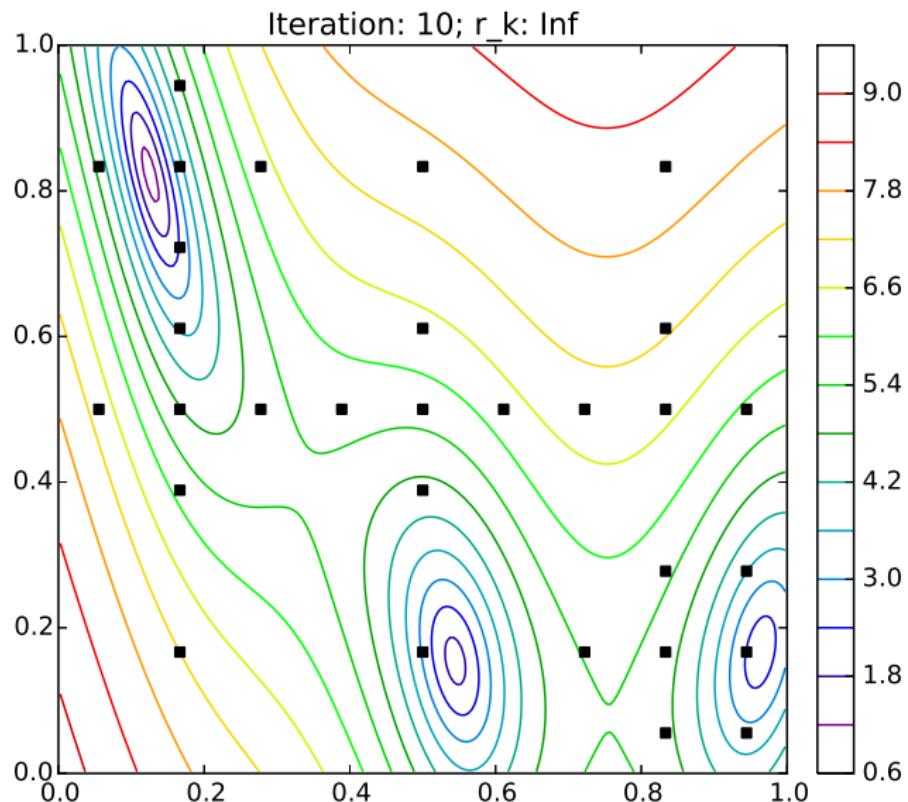


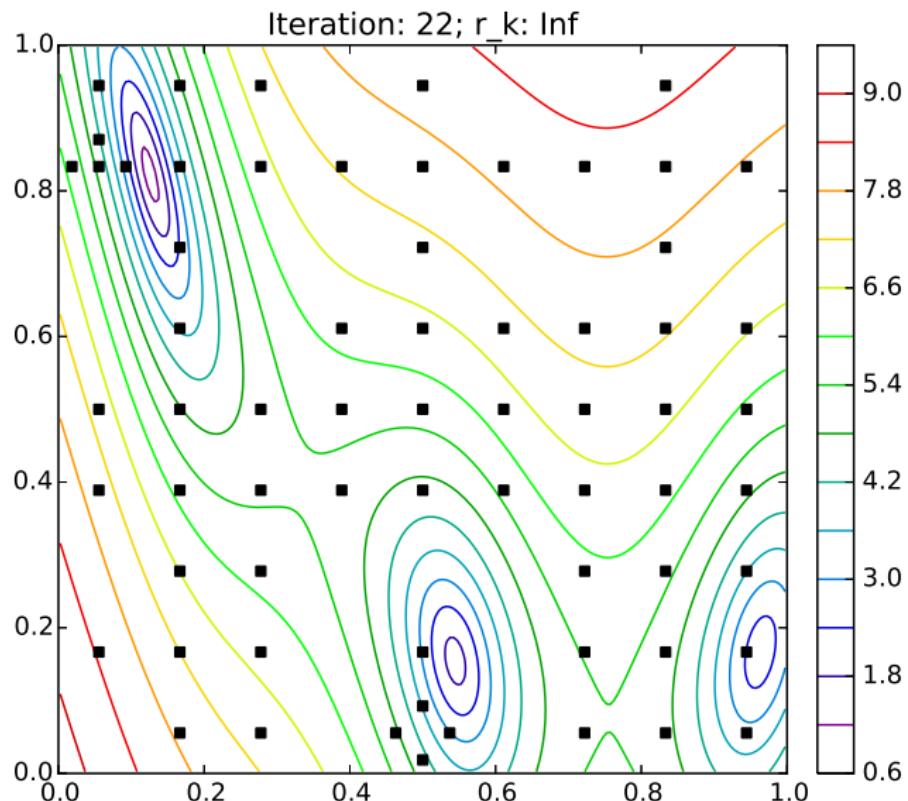


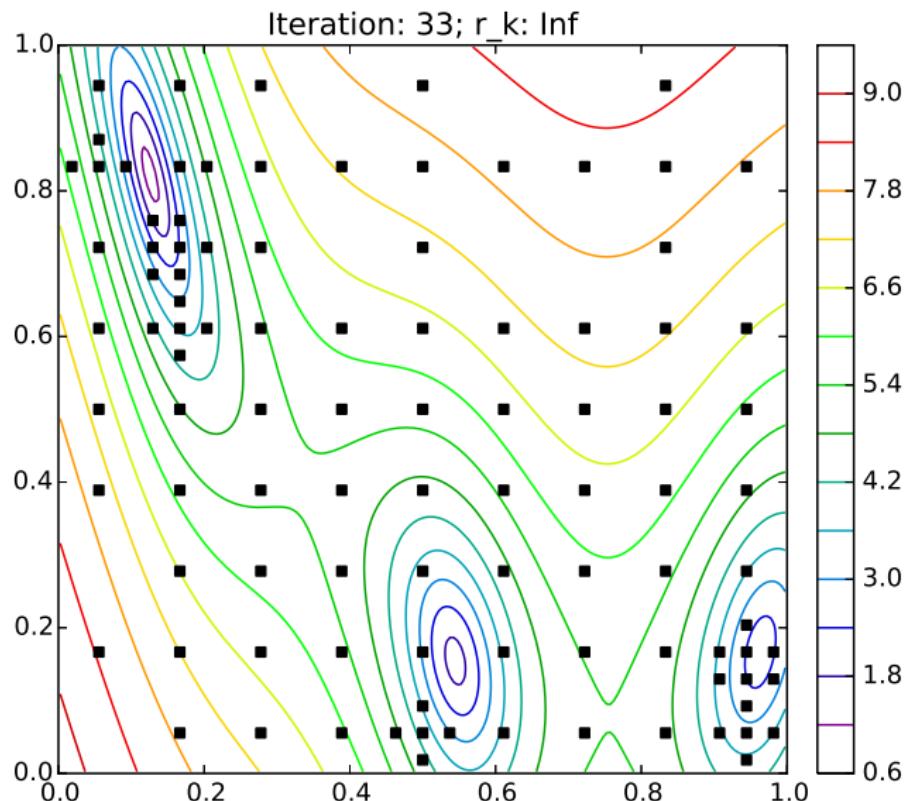












# Multistart Methods

Given some local optimization routine  $\mathcal{L}$ :

---

## Algorithm 1: General Multistart

---

**for**  $k = 1, 2, \dots$  **do**

    Evaluate  $f$  at  $N$  points drawn from  $\mathcal{D}$ .

    Start  $\mathcal{L}$  at some set (possibly empty) of previously evaluated points.

**end**

---

- ▶ Want to only start one run for each minima



# Multistart Methods

Given some local optimization routine  $\mathcal{L}$ :

---

## Algorithm 2: General Multistart

---

**for**  $k = 1, 2, \dots$  **do**

    Evaluate  $f$  at  $N$  points drawn from  $\mathcal{D}$ .

    Start  $\mathcal{L}$  at some set (possibly empty) of previously evaluated points.

**end**

---

- ▶ Want to only start one run for each minima
  
- ▶ Get to use problem specific local optimization routines
- ▶ Allows for many levels of parallelism (objective, local method, global method), but  $\mathcal{L}$  may involve many sequential evaluations of  $f$ ...



# Multistart Methods

Given some local optimization routine  $\mathcal{L}$ :

---

### Algorithm 3: General Multistart

---

```
for  $k = 1, 2, \dots$  do
    Evaluate  $f$  at  $N$  points drawn from  $\mathcal{D}$ .
    Start  $\mathcal{L}$  at some set (possibly empty) of previously evaluated points.
end
```

---

- ▶ Want to only start one run for each minima
- ▶ Get to use problem specific local optimization routines
- ▶ Allows for many levels of parallelism (objective, local method, global method), but  $\mathcal{L}$  may involve many sequential evaluations of  $f$ ...
- ▶ Which points should start runs?
- ▶ If resources are limited, how should points from each run receive priority?

# Multi-Level Single Linkage

Given some local optimization routine  $\mathcal{L}$ :

---

## Algorithm 4: MLSL

---

**for**  $k = 1, 2, \dots$  **do**

Evaluate  $f$  at  $N$  random points drawn uniformly from  $\mathcal{D}$ .

Start  $\mathcal{L}$  at any previously evaluated point  $x$ :

- ▶ that is not a local minima
- ▶  $\nexists x_i : \|x - x_i\| \leq r_k$  and  $f(x_i) < f(x)$

**end**

---

- ▶ Doesn't naturally translate when evaluations of  $f$  are limited
- ▶ Ignores some points when deciding where to start  $\mathcal{L}$

[Rinnooy Kan and Timmer, *Mathematical Programming*, 39(1):57–78, 1987]

## MLSL: (S2)–(S4)

$$\hat{x} \in \mathcal{S}_k$$

- (S2)  $\nexists x \in \mathcal{S}_k$  *with*  
 $[\|\hat{x} - x\| \leq r_k \text{ and } f(x) < f(\hat{x})]$
- (S3)  $\hat{x}$  *has not started a local optimization run*
- (S4)  $\hat{x}$  *is at least  $\mu$  from  $\partial\mathcal{D}$  and  $\nu$  from known local minima*



# BAMLM

## MLSL: (S2)–(S4)

$$\hat{x} \in \mathcal{S}_k$$

- (S1)  $\nexists x \in \mathcal{L}_k$  with  
 $[\|\hat{x} - x\| \leq r_k \text{ and } f(x) < f(\hat{x})]$
- (S2)  $\nexists x \in \mathcal{S}_k$  with  
 $[\|\hat{x} - x\| \leq r_k \text{ and } f(x) < f(\hat{x})]$
- (S3)  $\hat{x}$  has not started a local optimization run
- (S4)  $\hat{x}$  is at least  $\mu$  from  $\partial\mathcal{D}$  and  $\nu$  from known local minima

## BAMLM: (S1)–(S4), (L1)–(L6)

$$\hat{x} \in \mathcal{L}_k$$

- (L1)  $\nexists x \in \mathcal{L}_k$   
 $[\|\hat{x} - x\| \leq r_k \text{ and } f(x) < f(\hat{x})]$
- (L2)  $\nexists x \in \mathcal{S}_k$  with  
 $[\|\hat{x} - x\| \leq r_k \text{ and } f(x) < f(\hat{x})]$
- (L3)  $\hat{x}$  has not started a local optimization run
- (L4)  $\hat{x}$  is at least  $\mu$  from  $\partial\mathcal{D}$  and  $\nu$  from known local minima
- (L5)  $\hat{x}$  is not in an active local optimization run and has not been ruled stationary
- (L6)  $\exists r_k$ -descent path in  $\mathcal{H}_k$  from some  $x \in \mathcal{S}_k$  satisfying (S2–S4) to  $\hat{x}$

## Multi-Level Single Linkage

- ▶ If  $f \in C^2$ , with local minima in the interior of  $\mathcal{D}$ , and the distance between these minima is bounded away from zero.
- ▶ If  $\mathcal{L}$  is strictly descent and converges to minimum (not stationary point).
- ▶

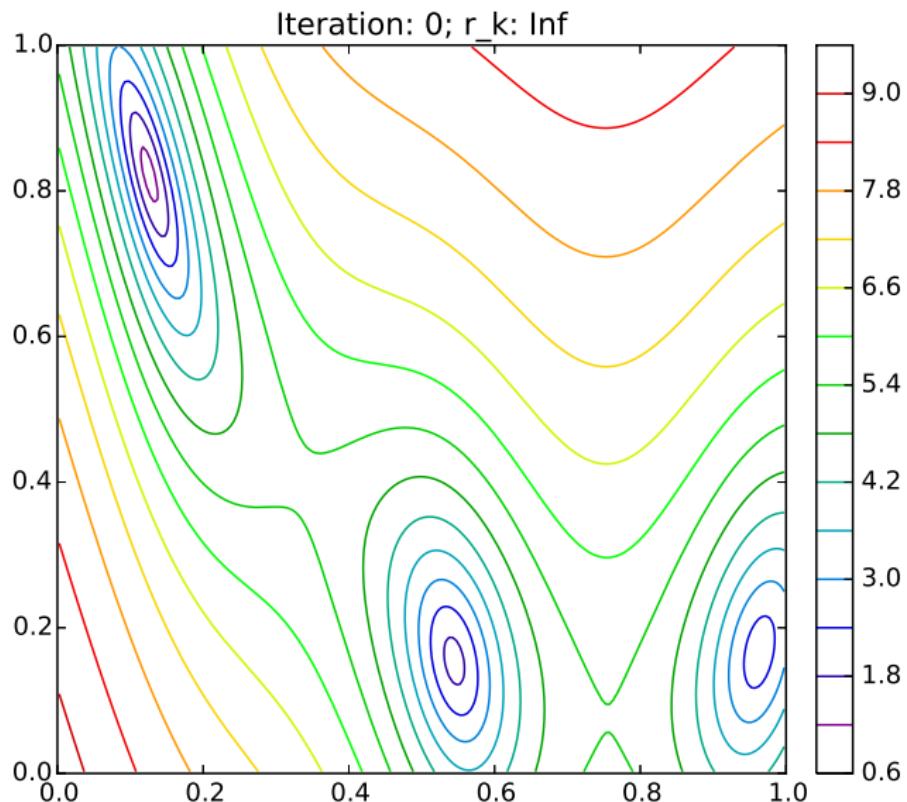
$$r_k = \frac{1}{\sqrt{\pi}} \sqrt[n]{\Gamma\left(1 + \frac{n}{2}\right) \text{vol}(\mathcal{D}) \frac{\sigma \log kN}{kN}} \quad (1)$$

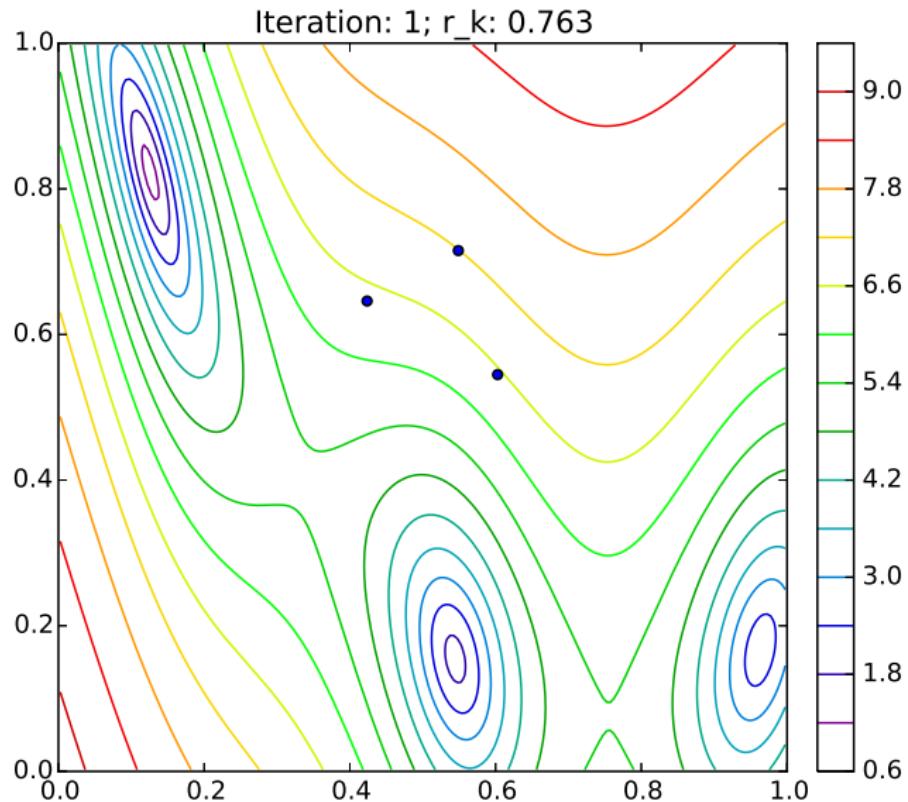
### Theorem

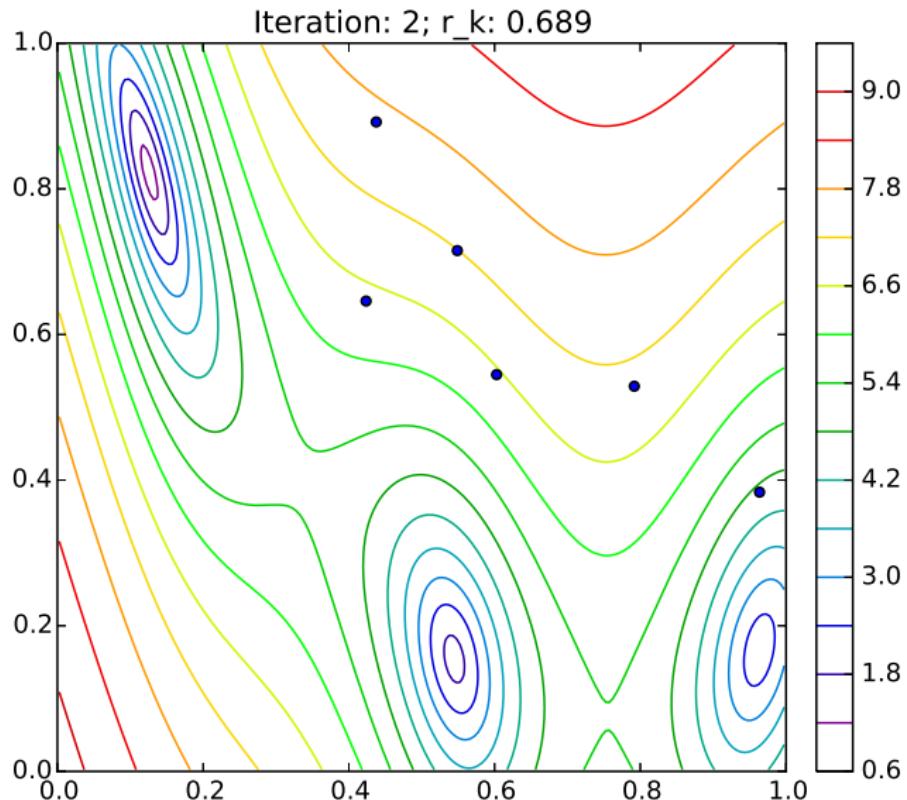
If  $r_k \rightarrow 0$ , all local minima will be found almost surely.

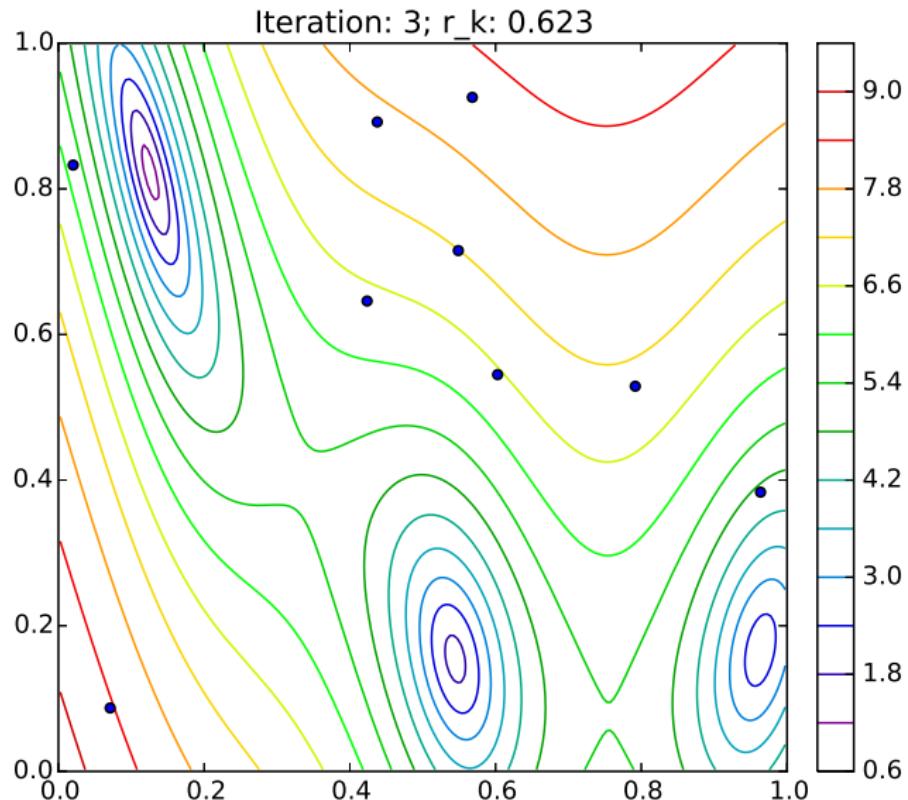
If  $r_k$  is defined by (1) with  $\sigma > 4$ , even if the sampling continues forever, the total number of local searches ever started is finite almost surely.

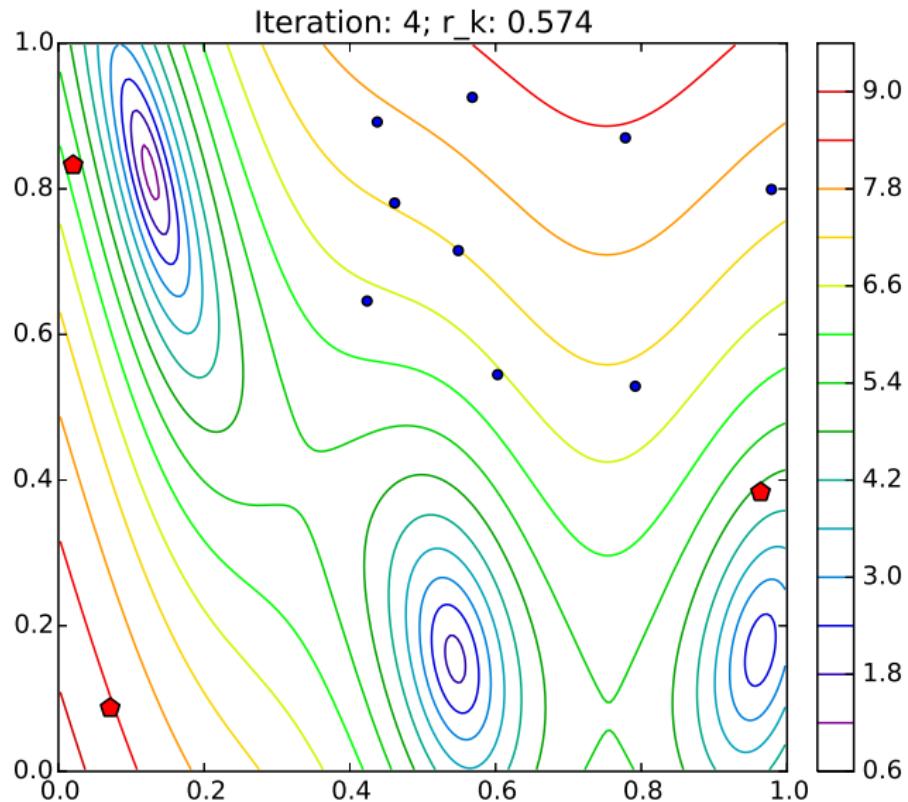


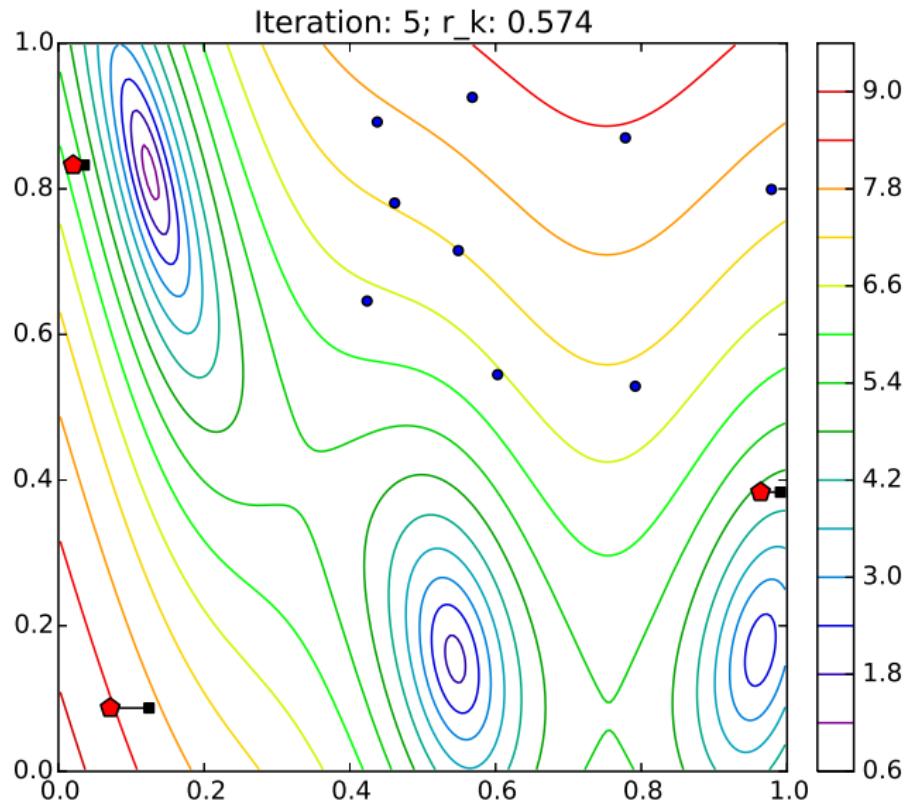


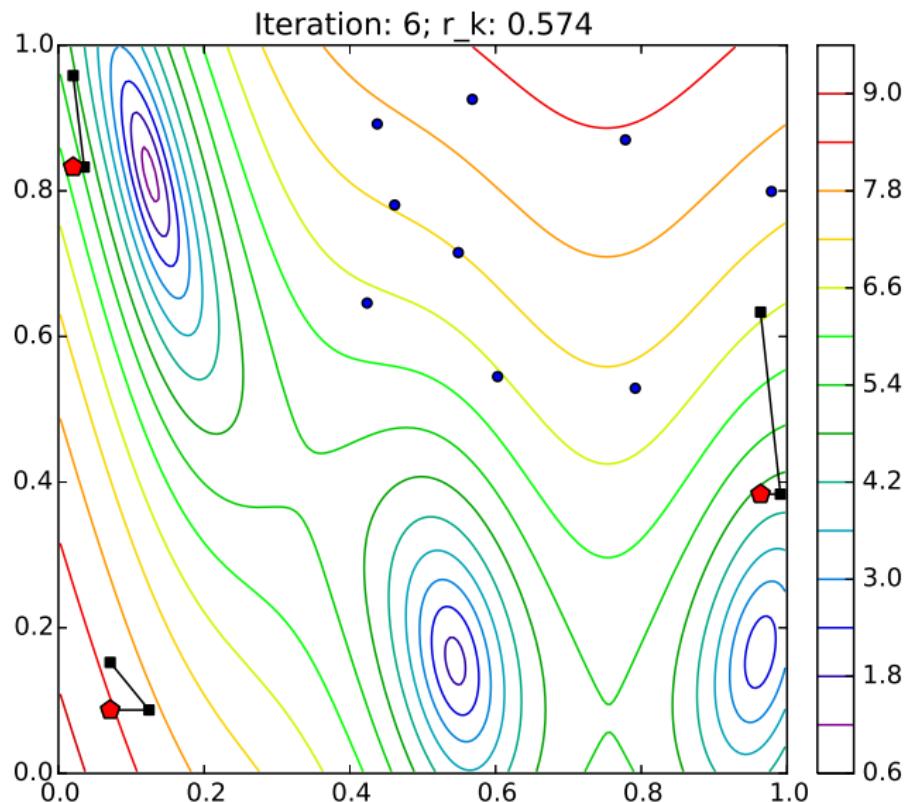


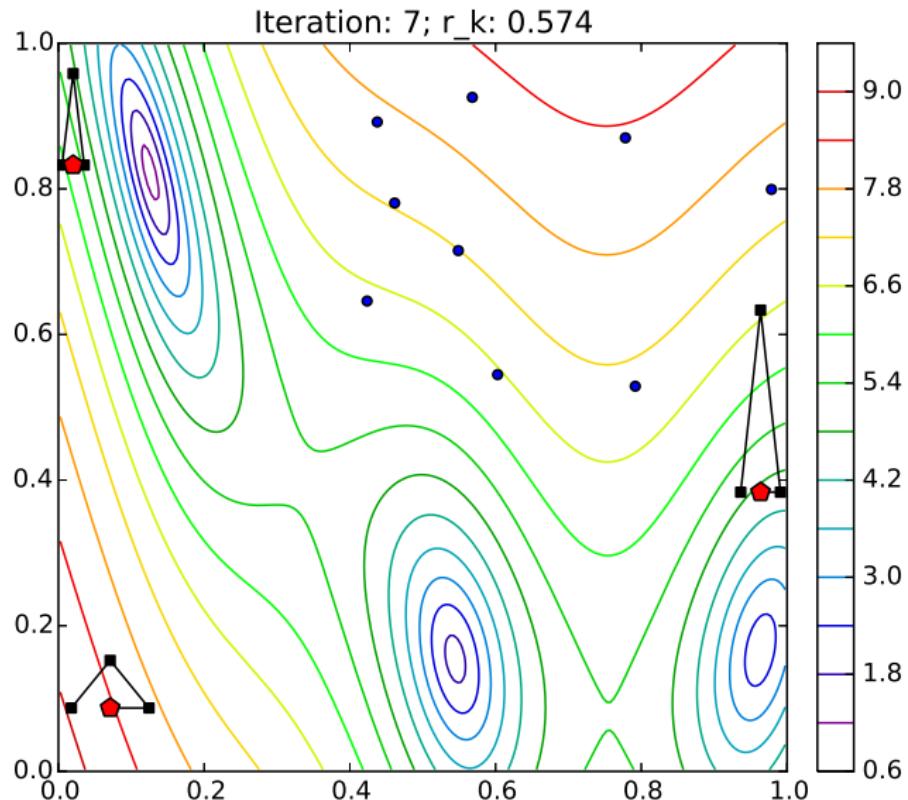


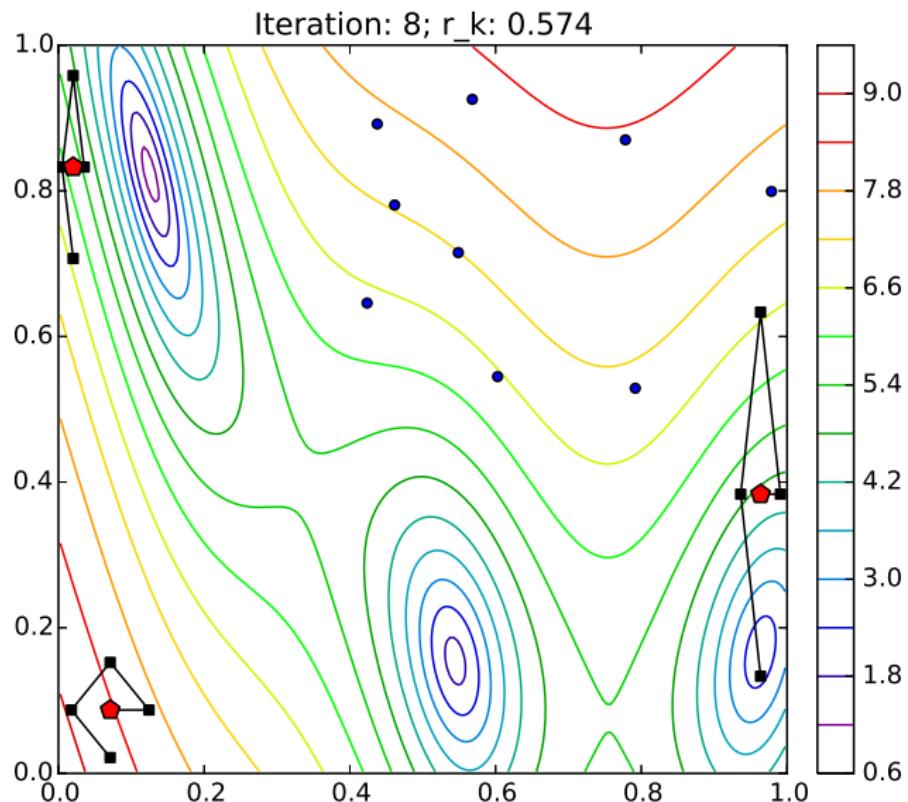


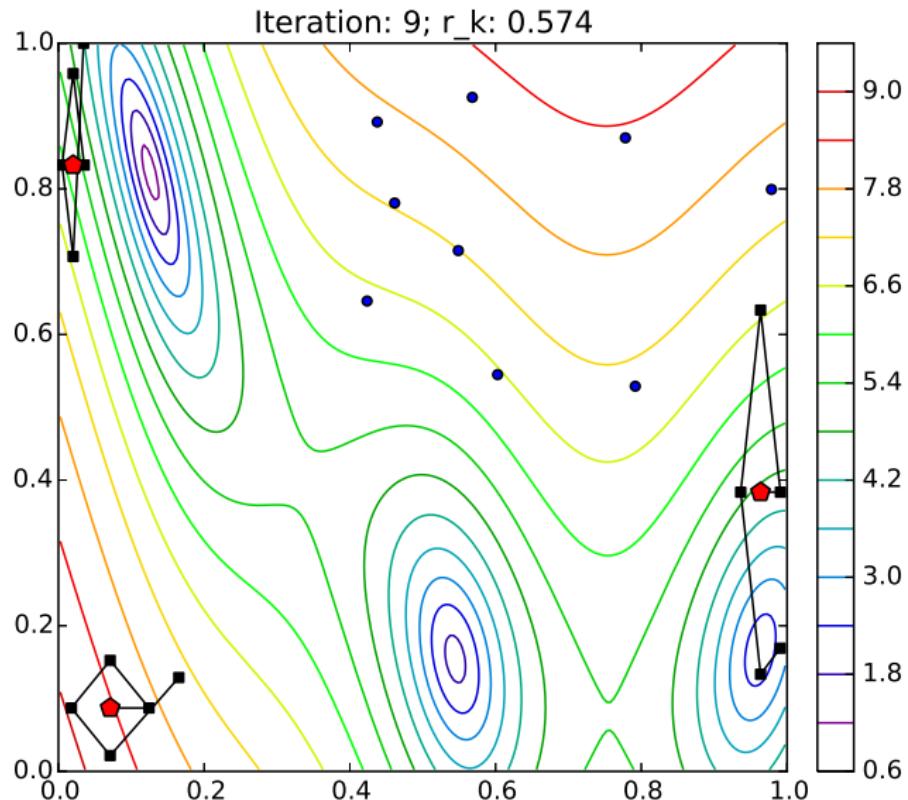


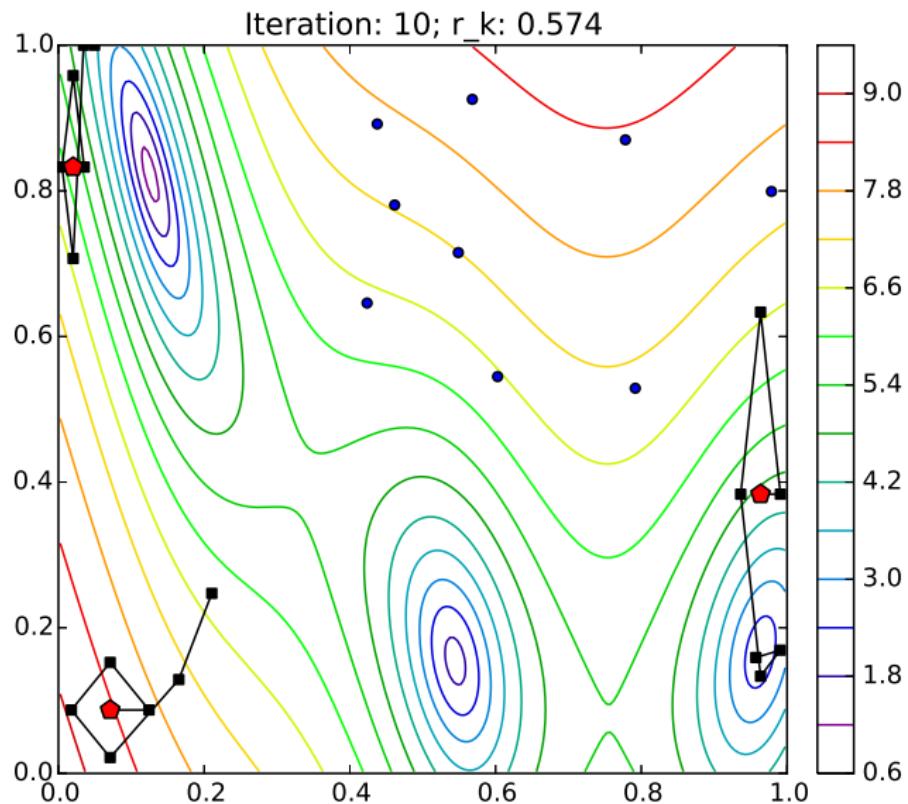


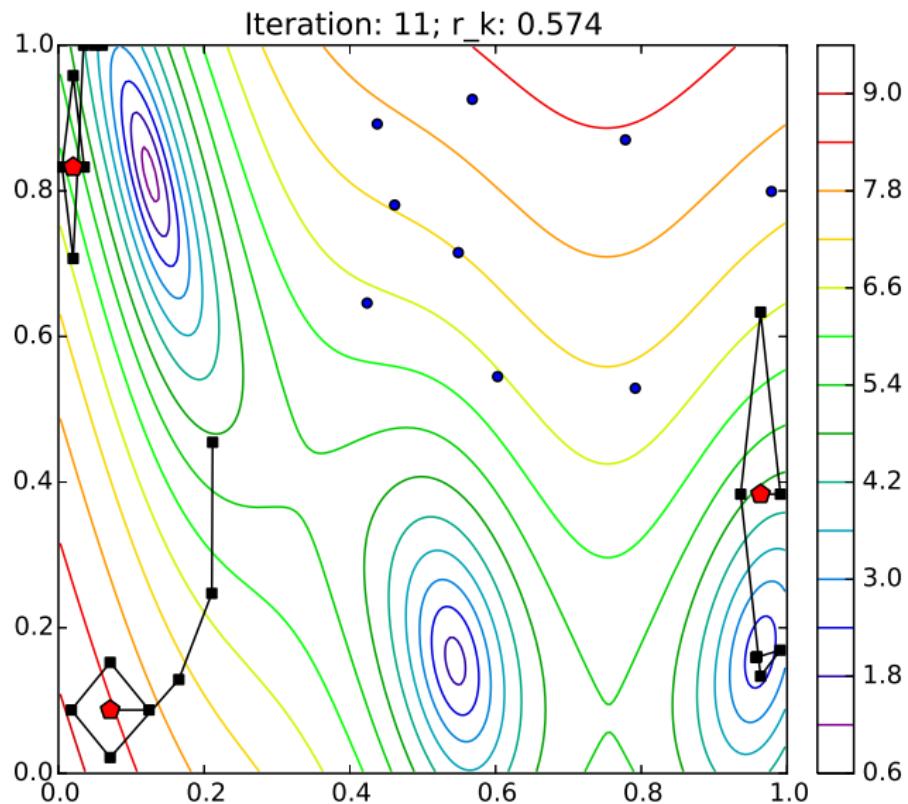


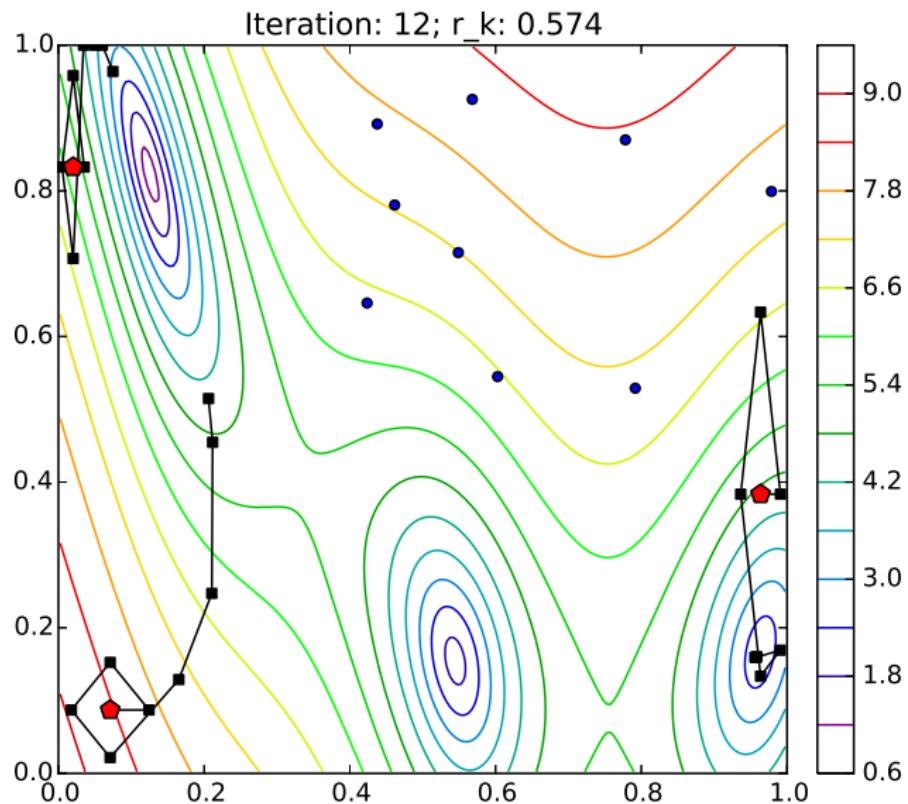


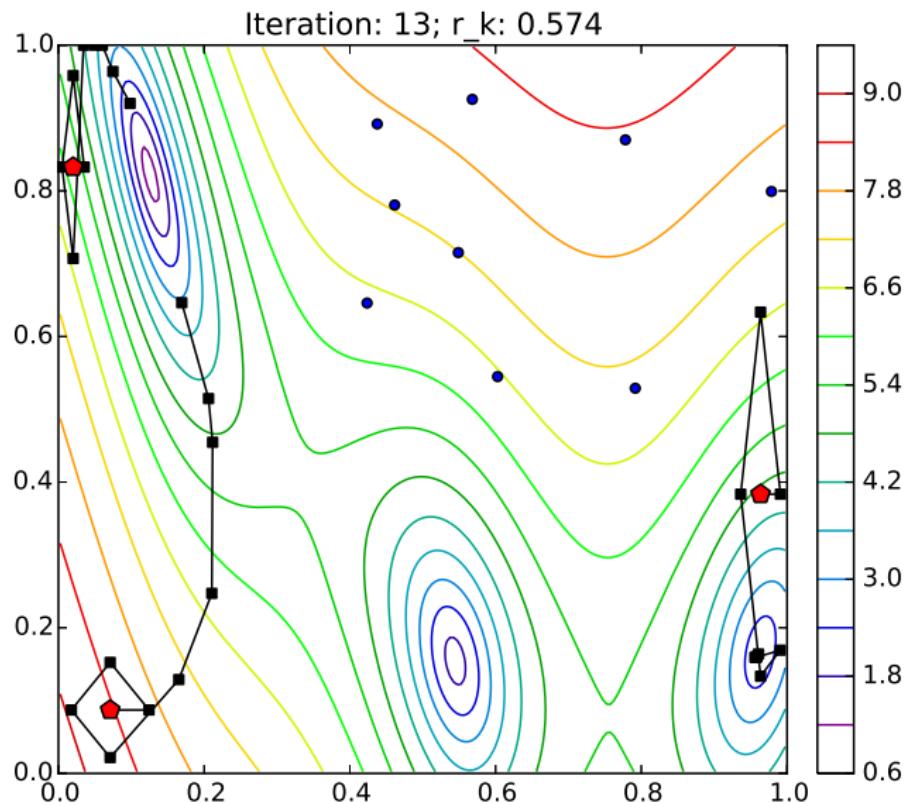


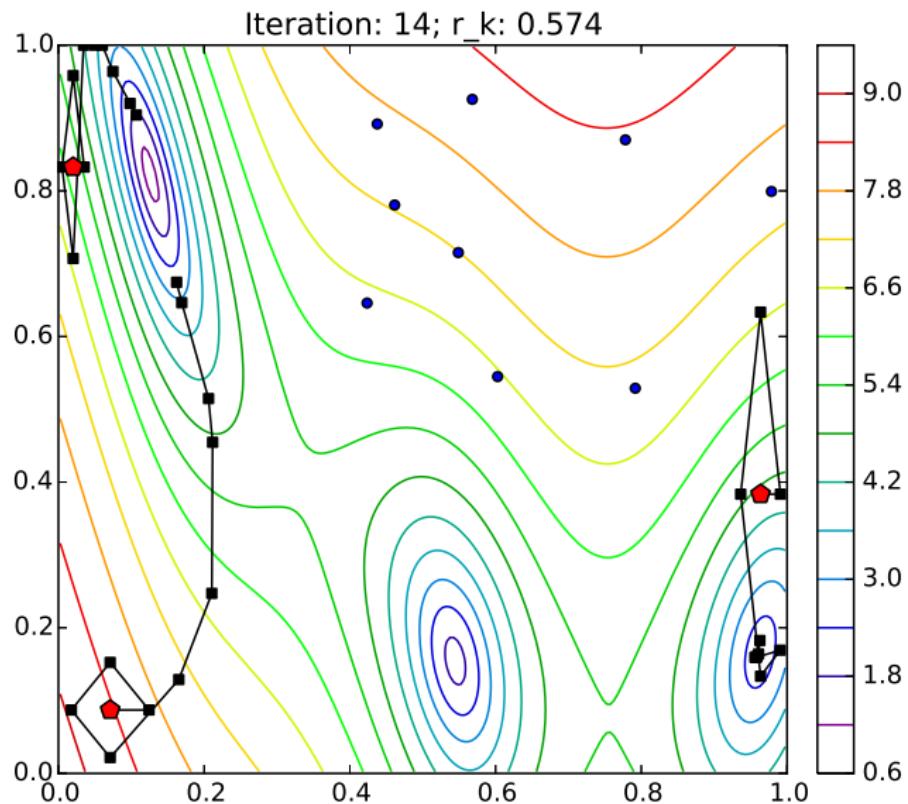


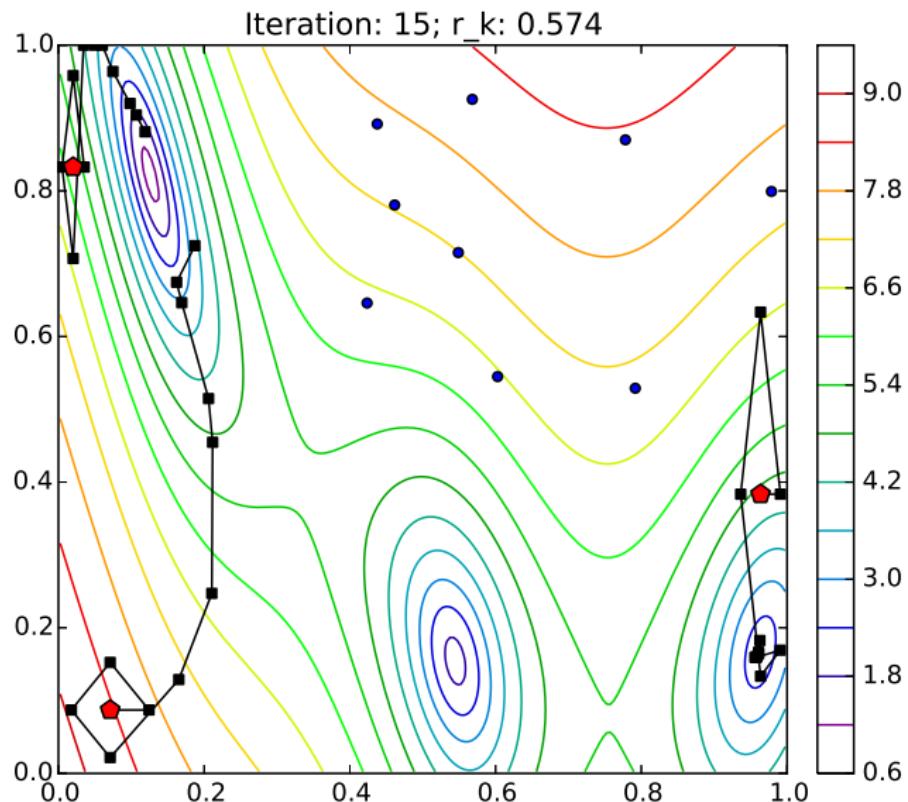


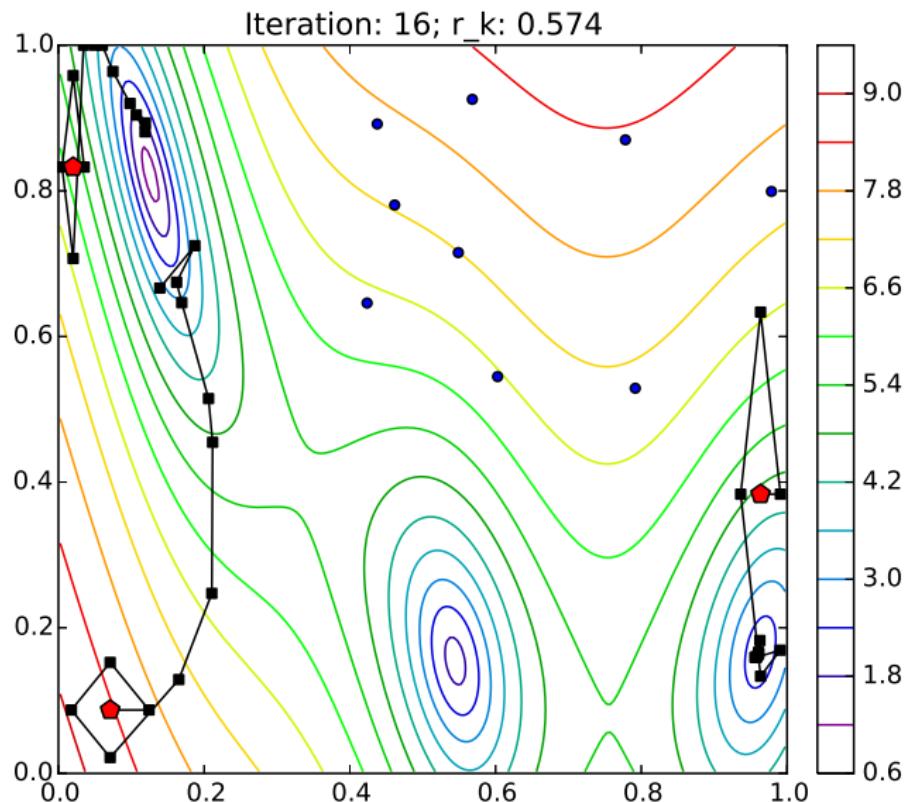


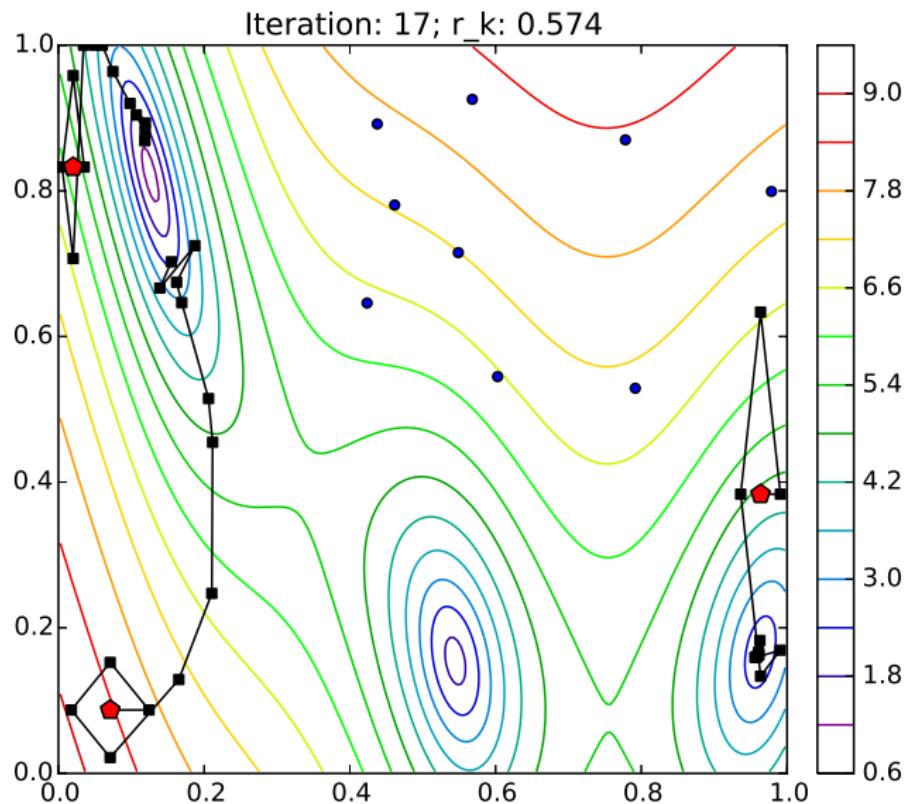


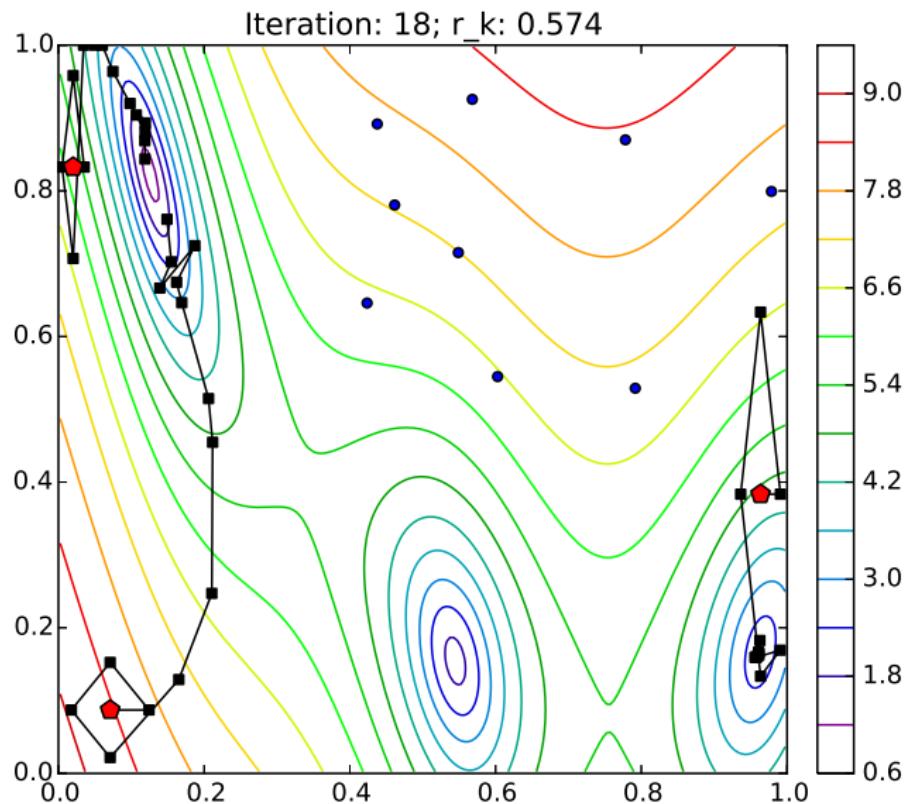


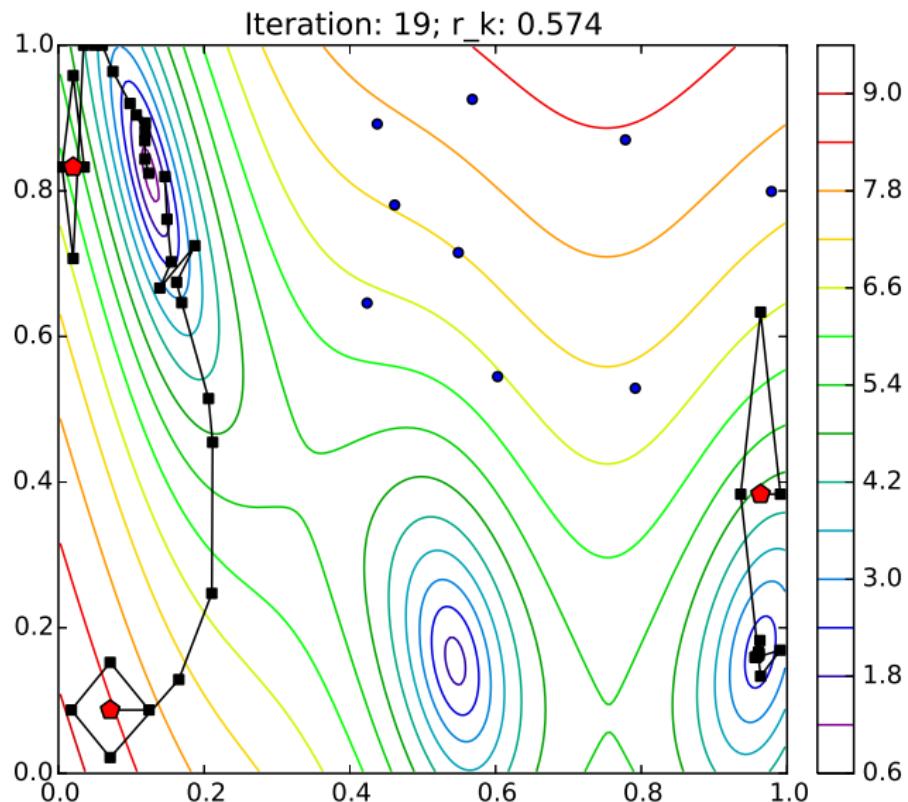


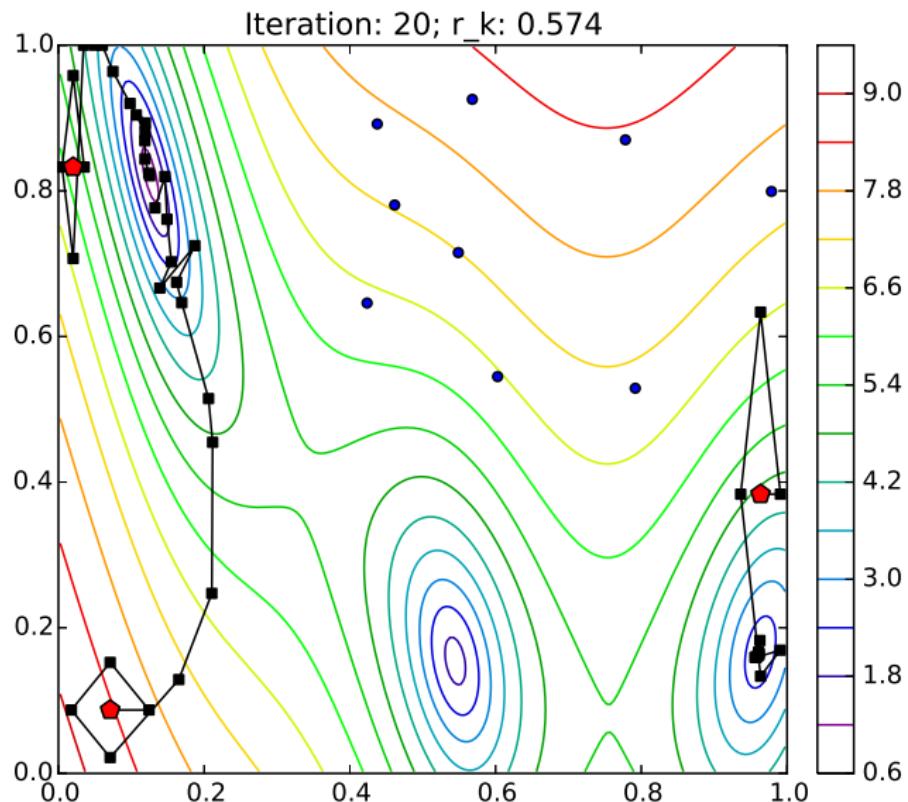


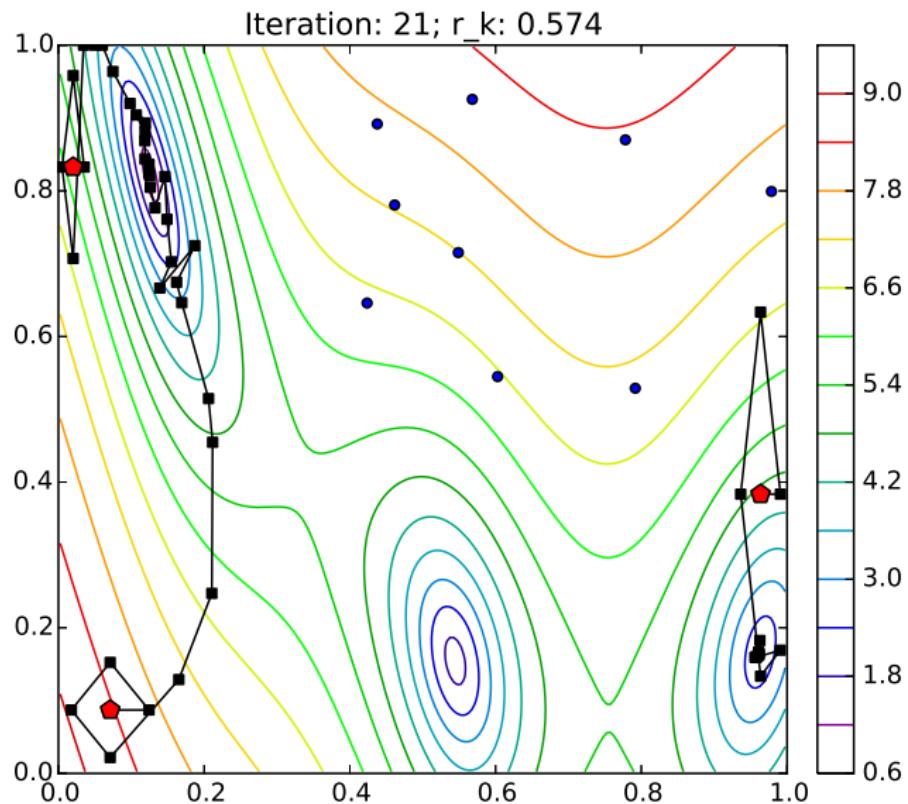


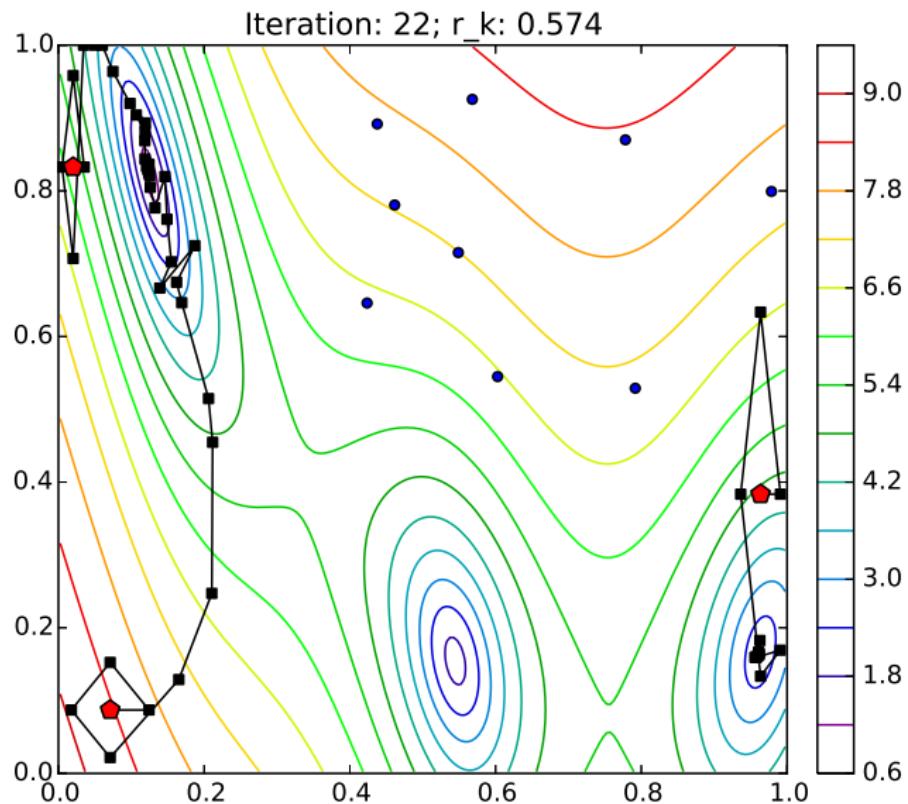


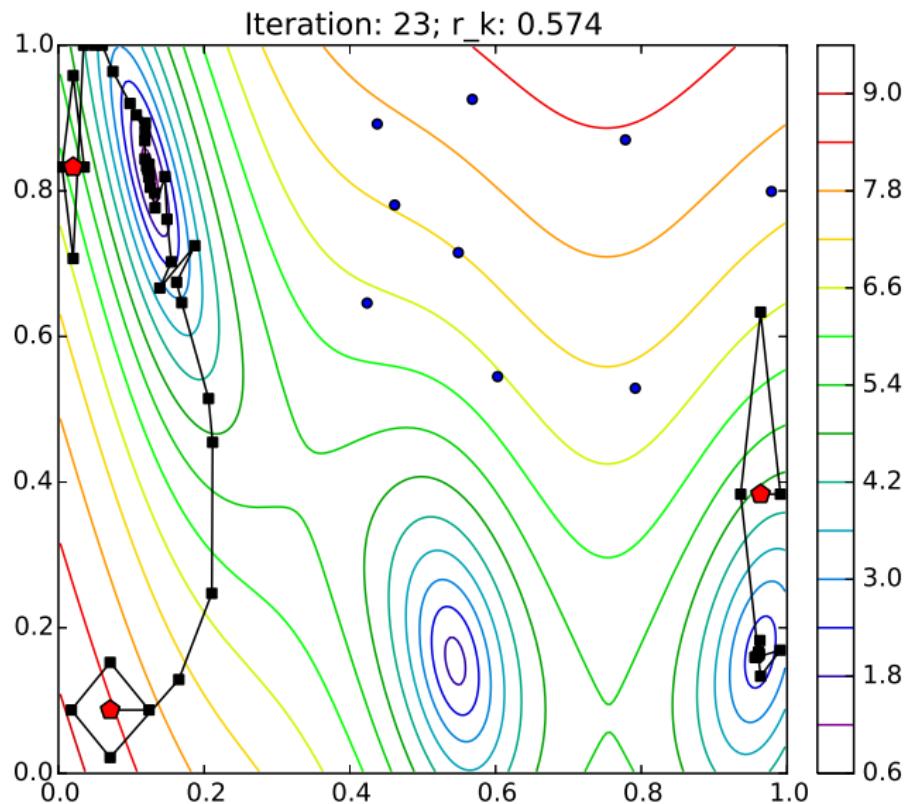


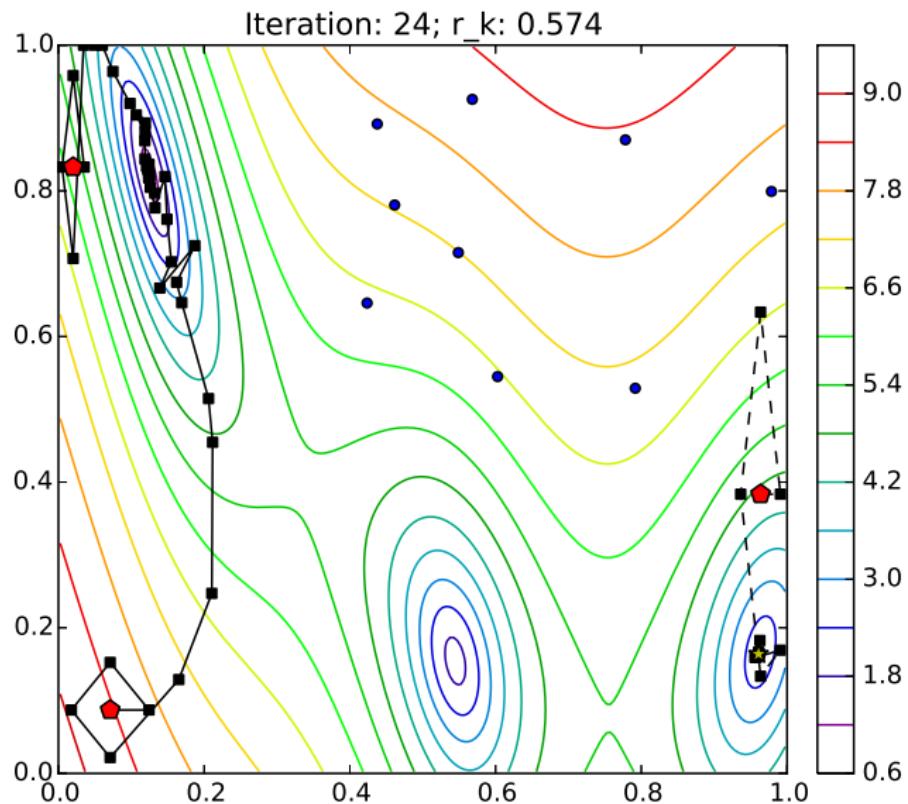


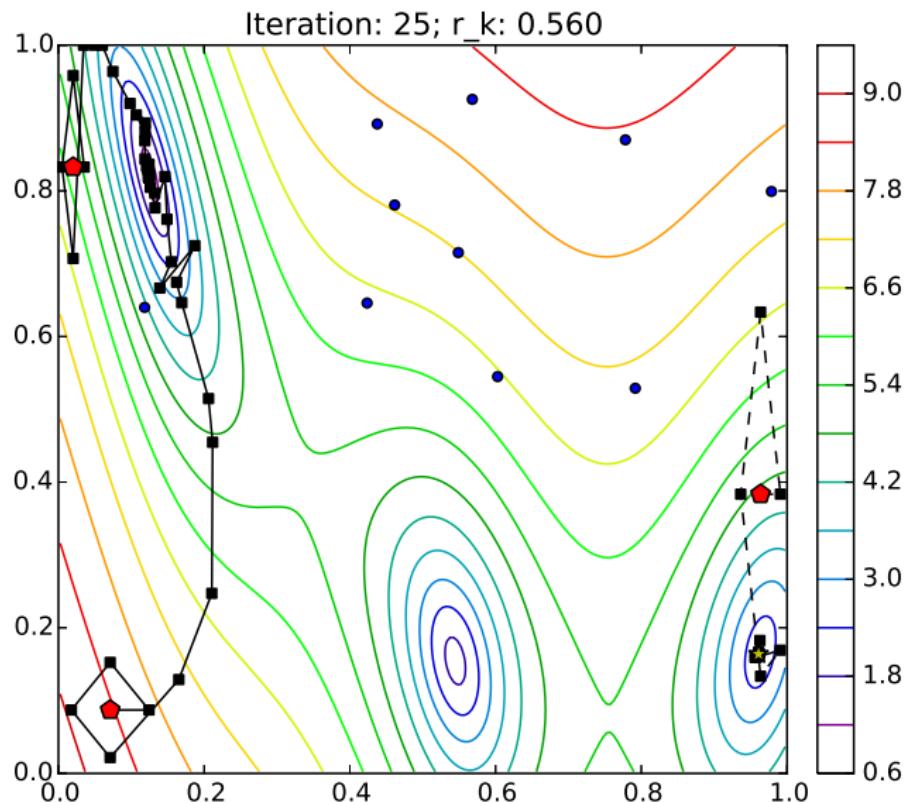


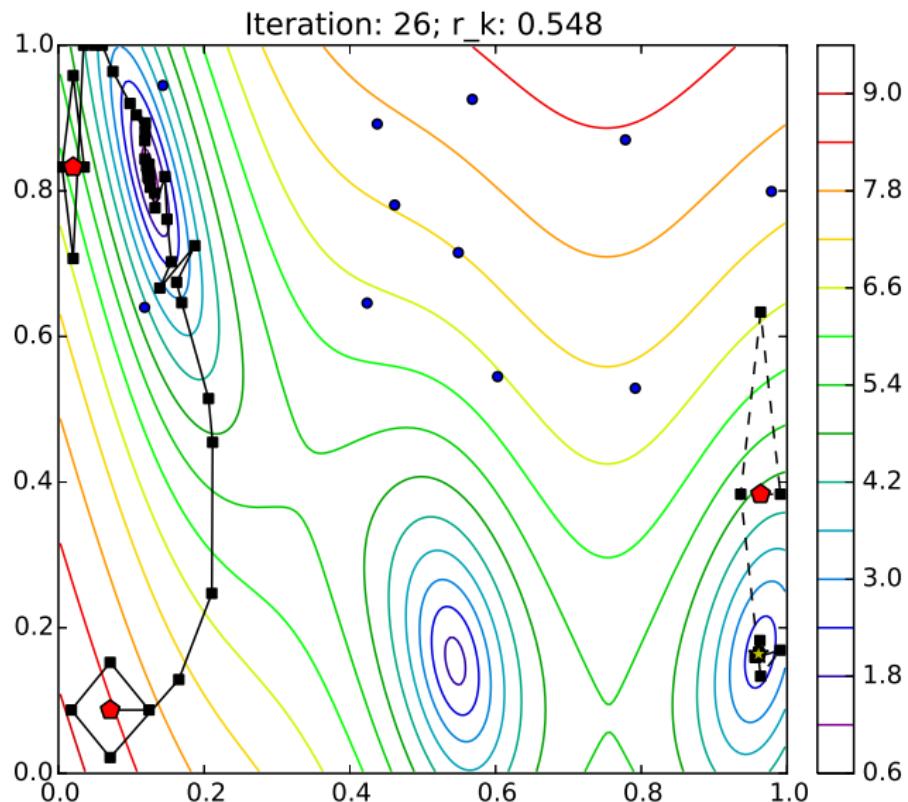


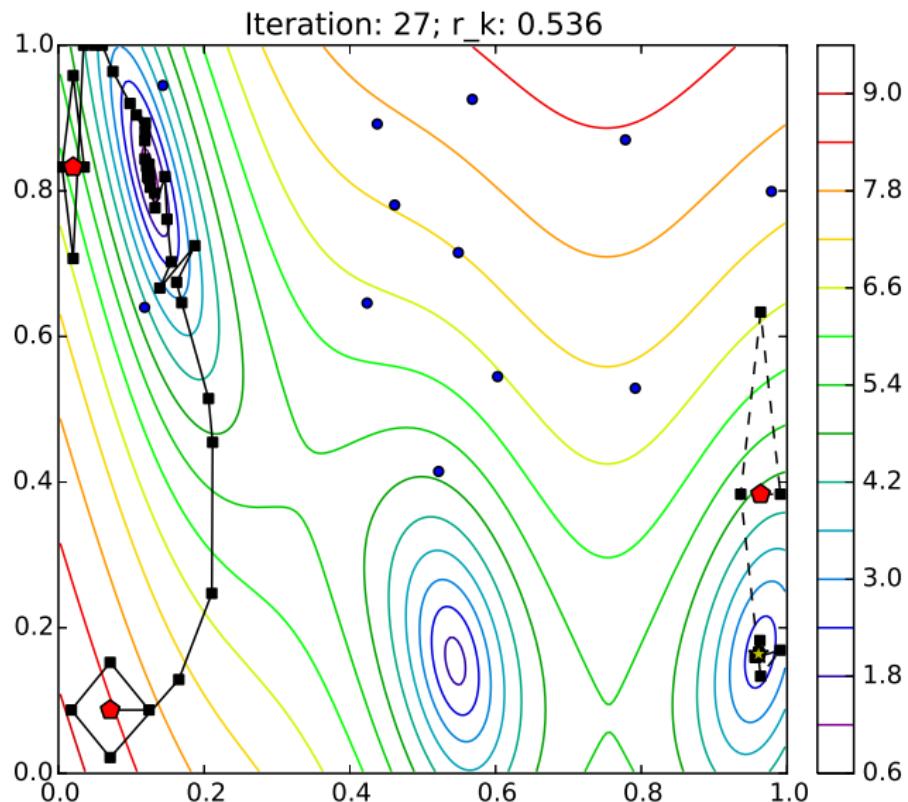


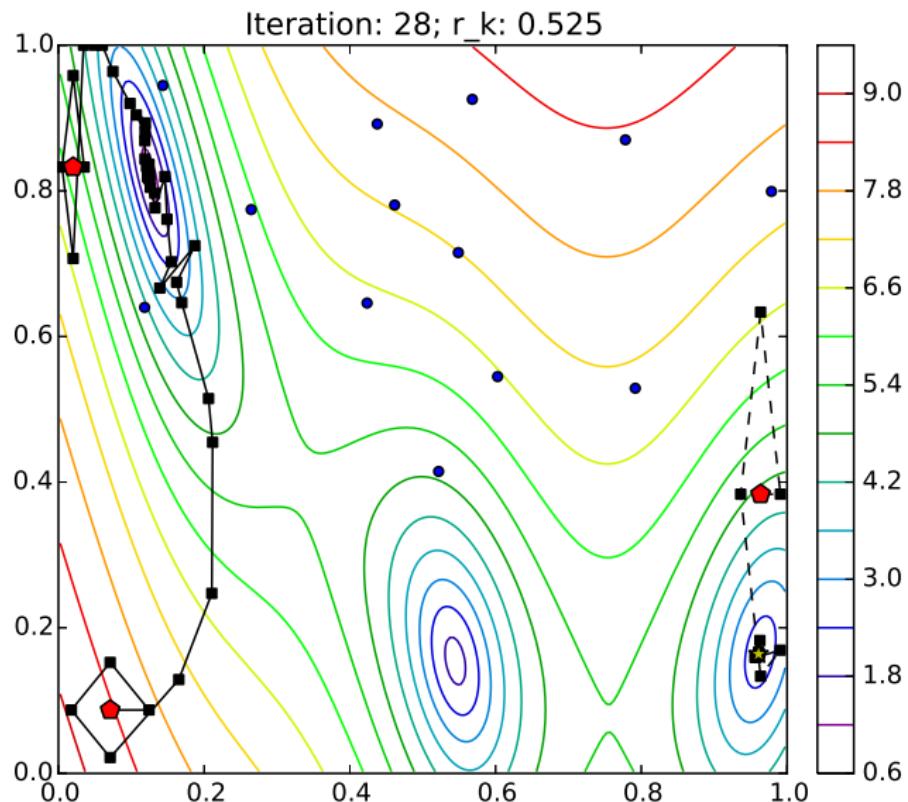


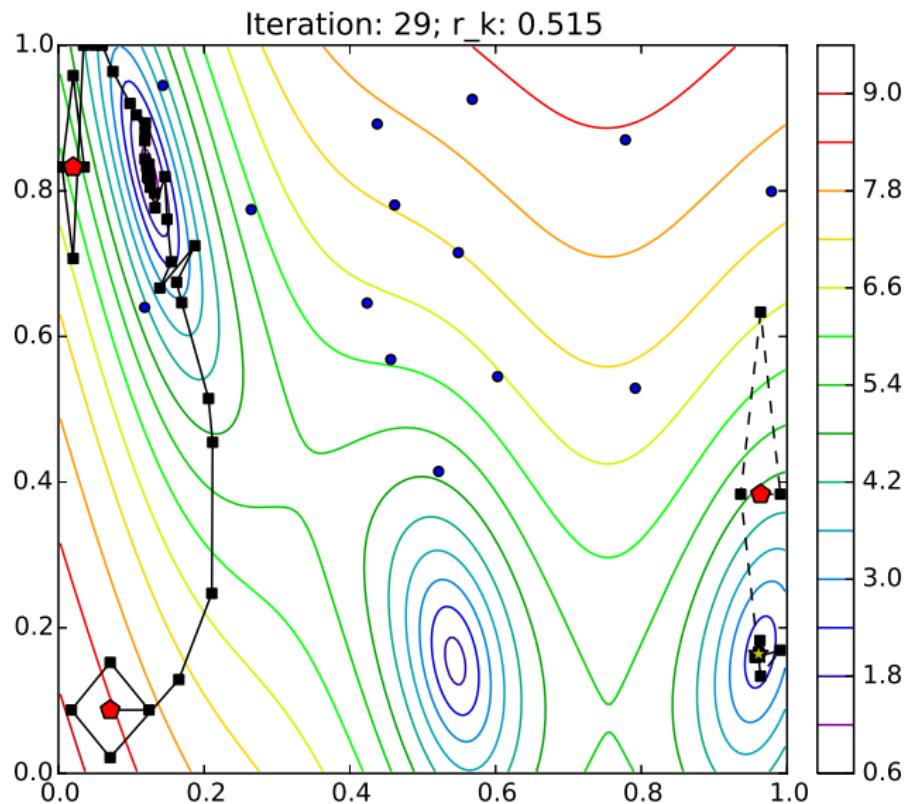


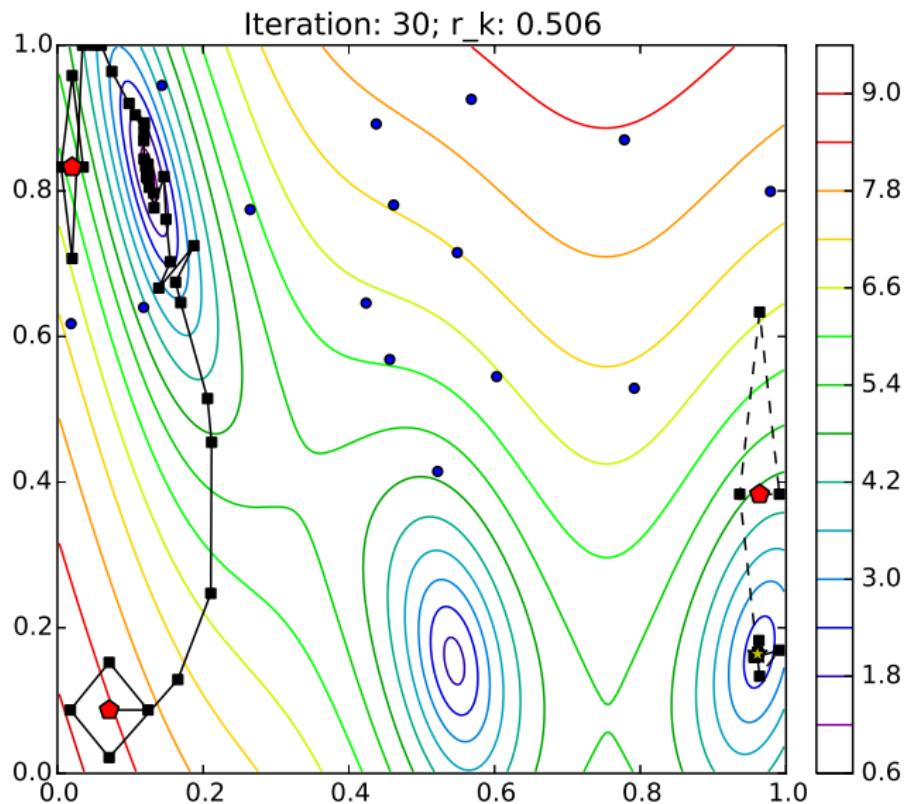


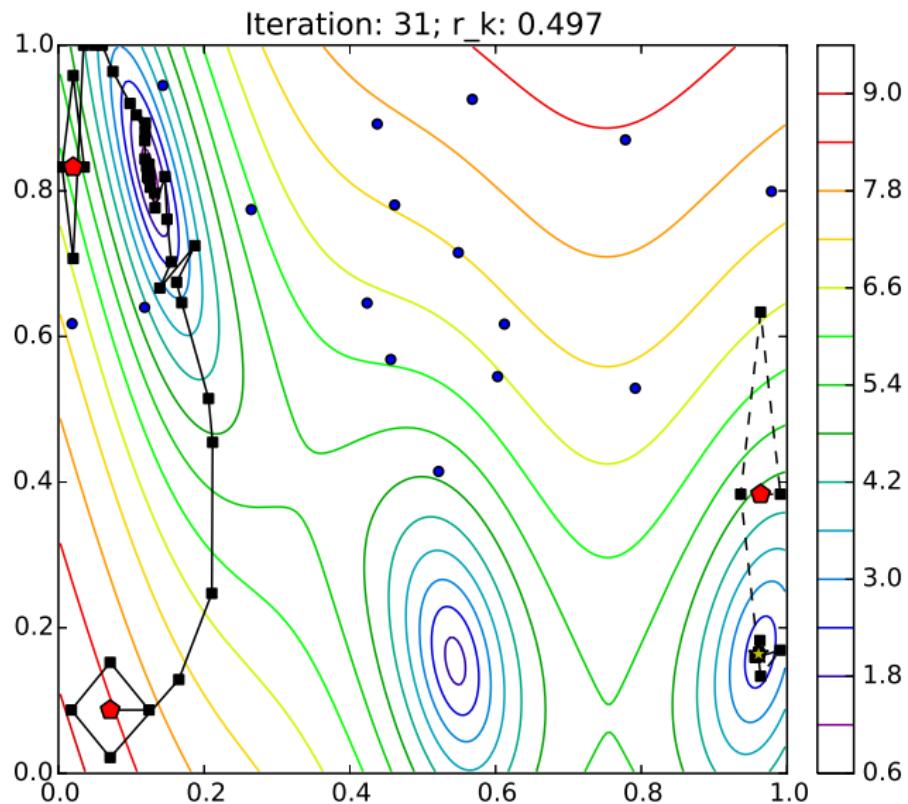


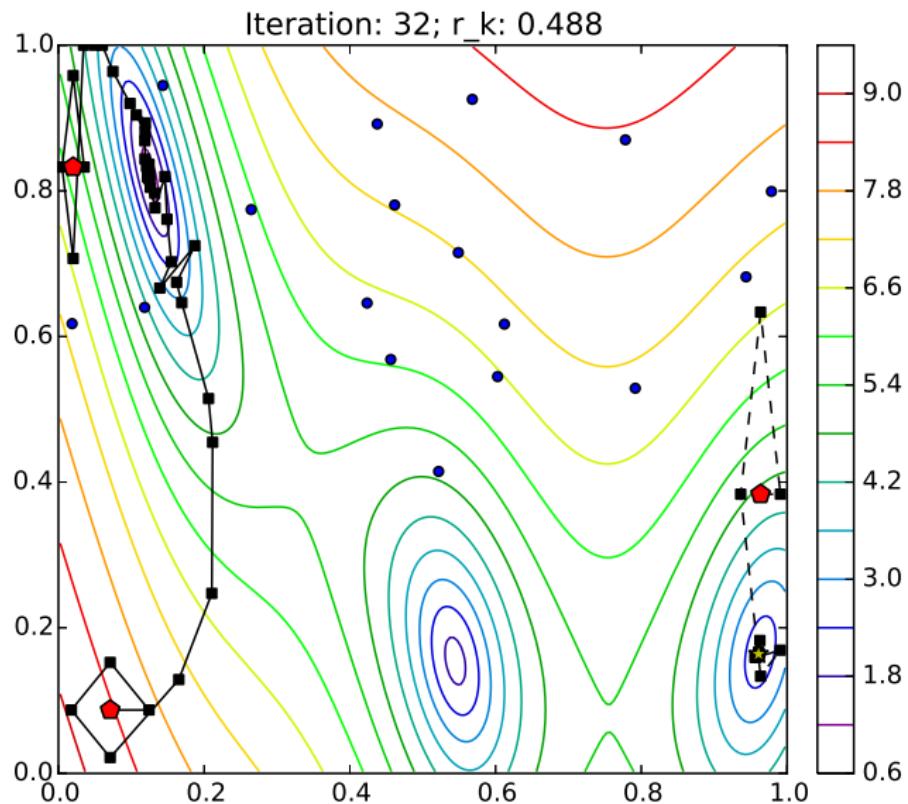


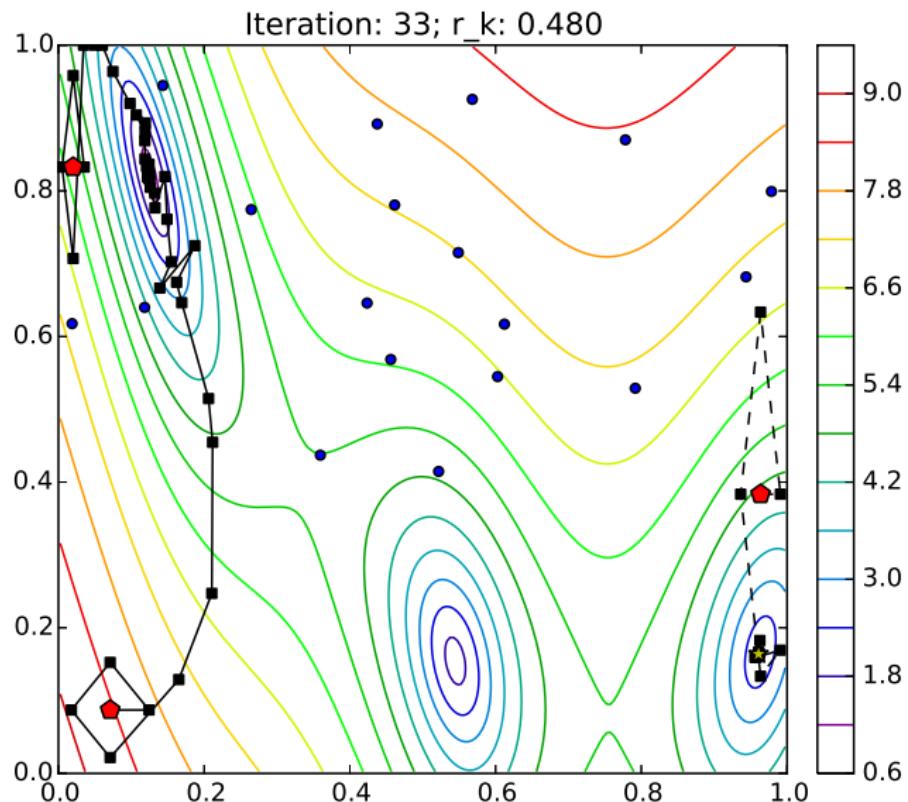


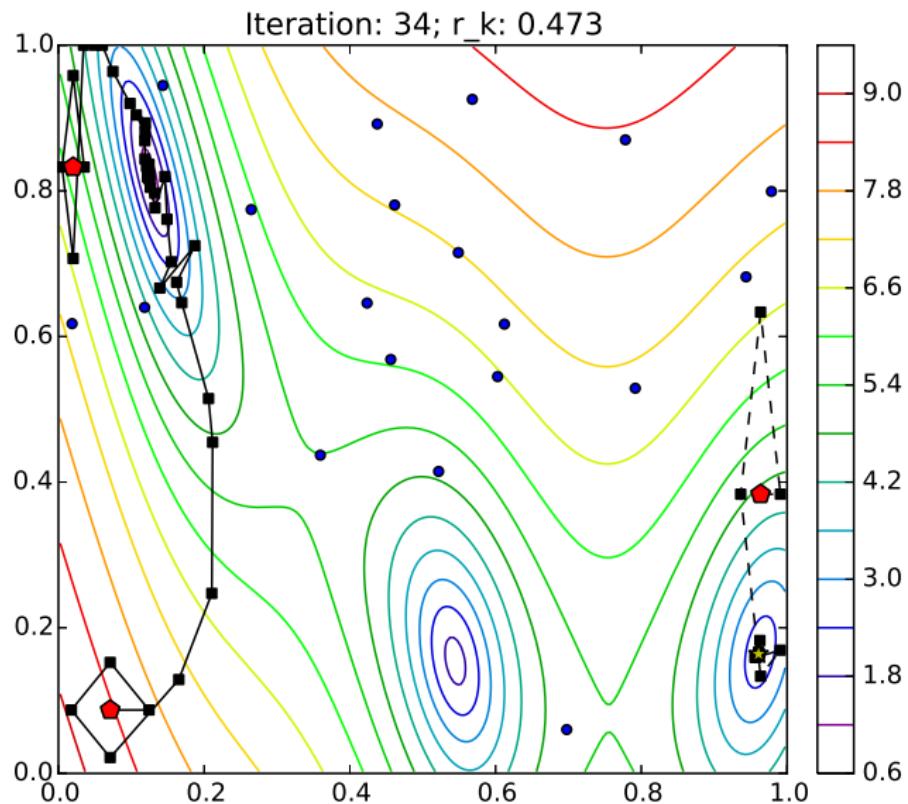


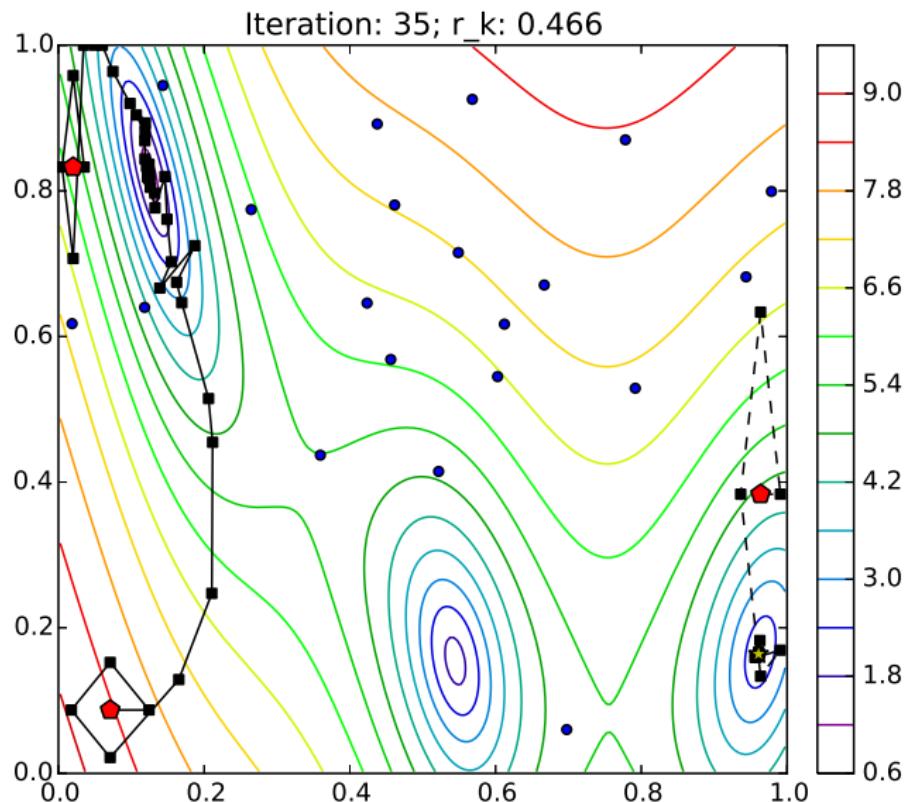


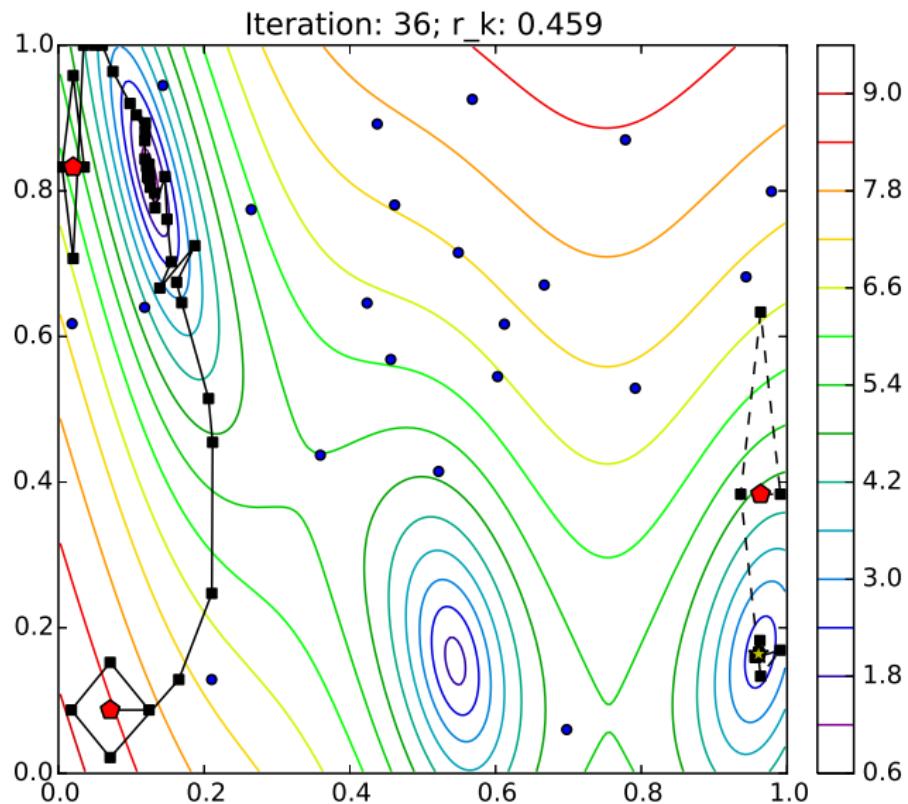


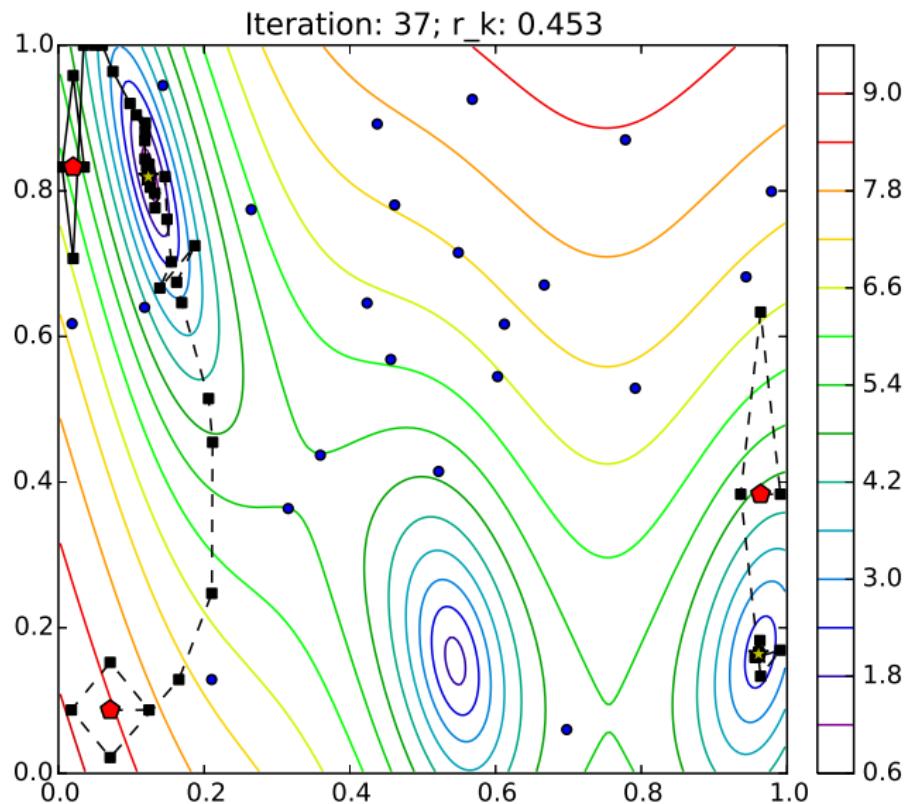


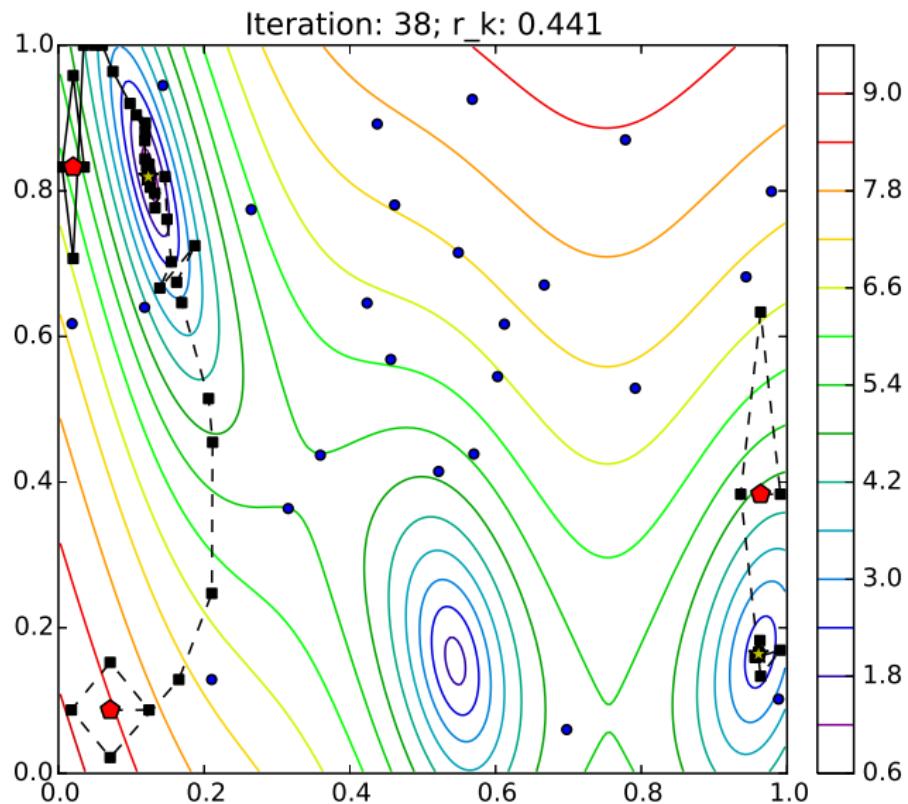


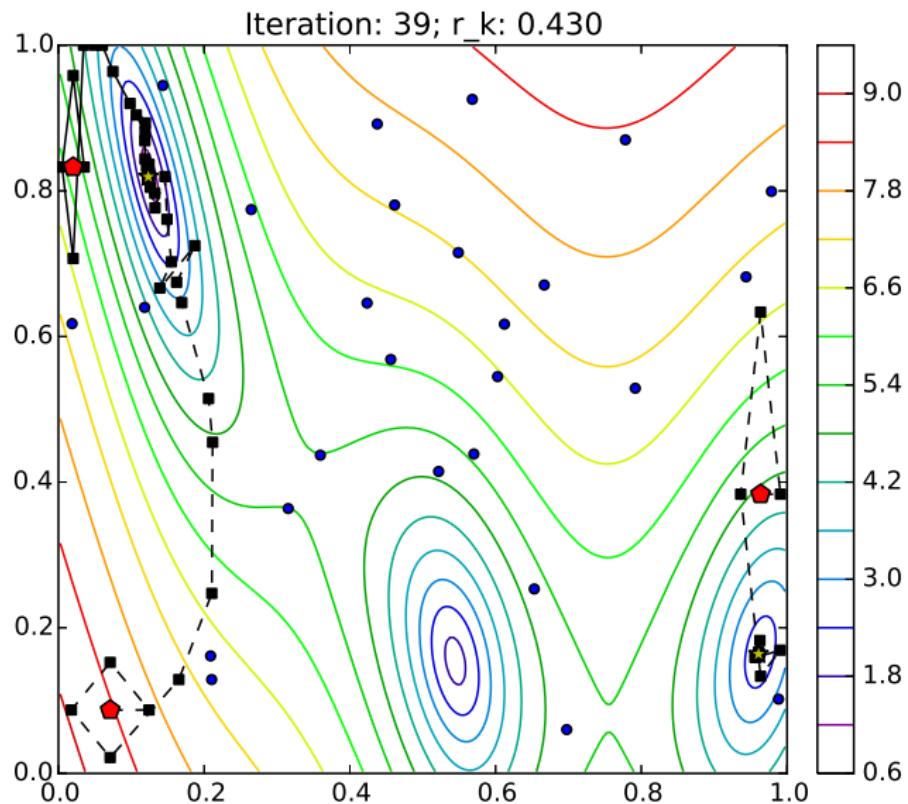


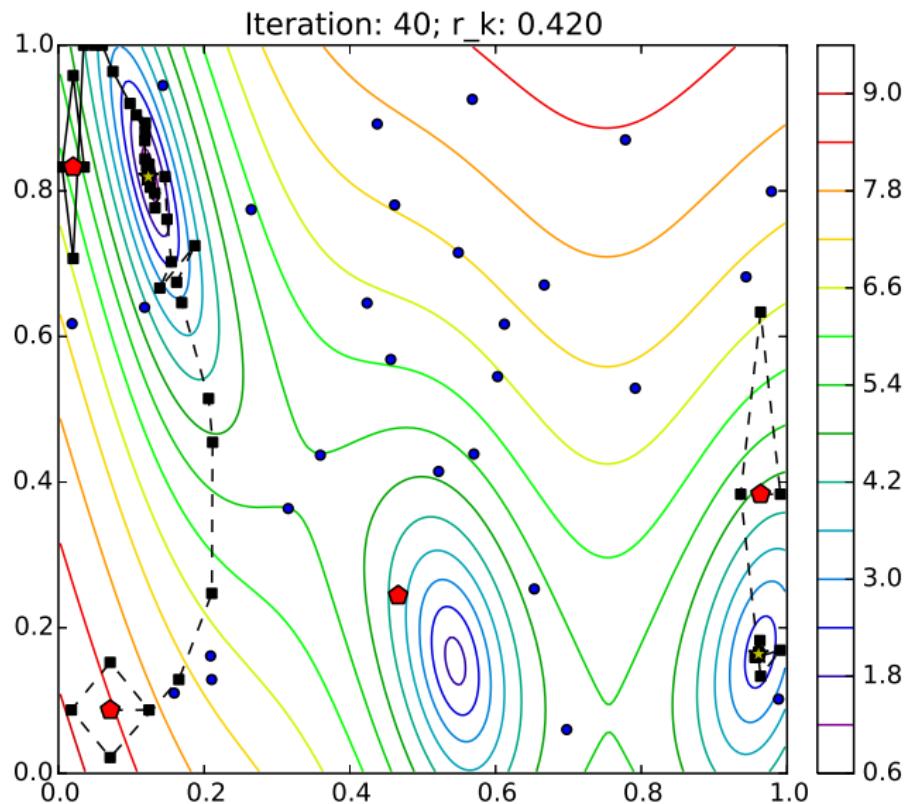


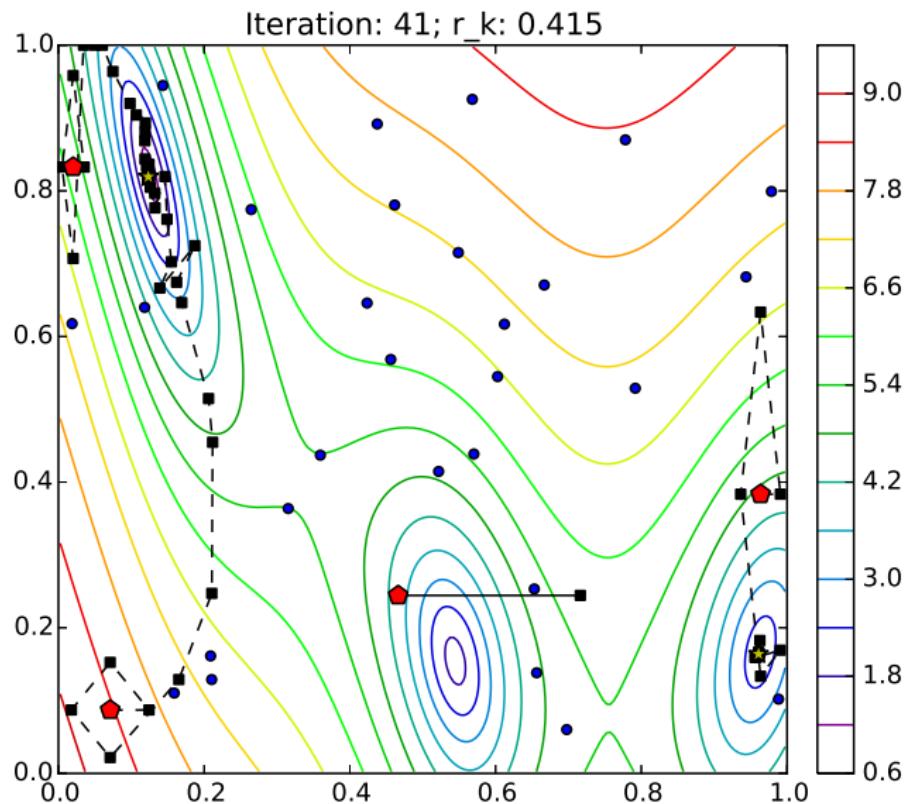


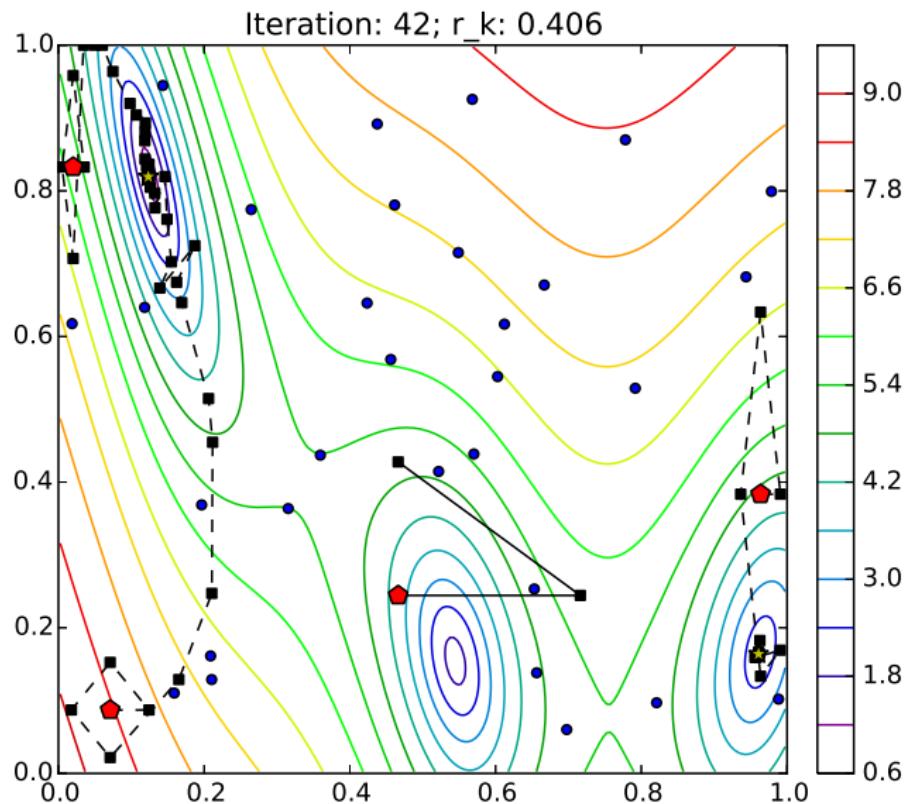


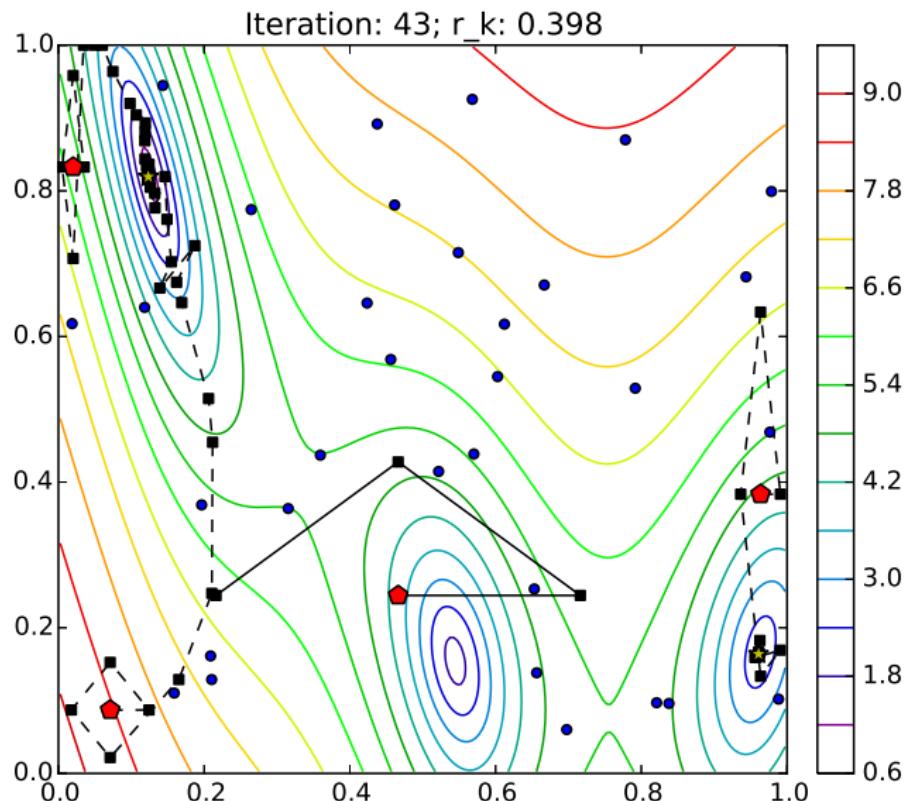


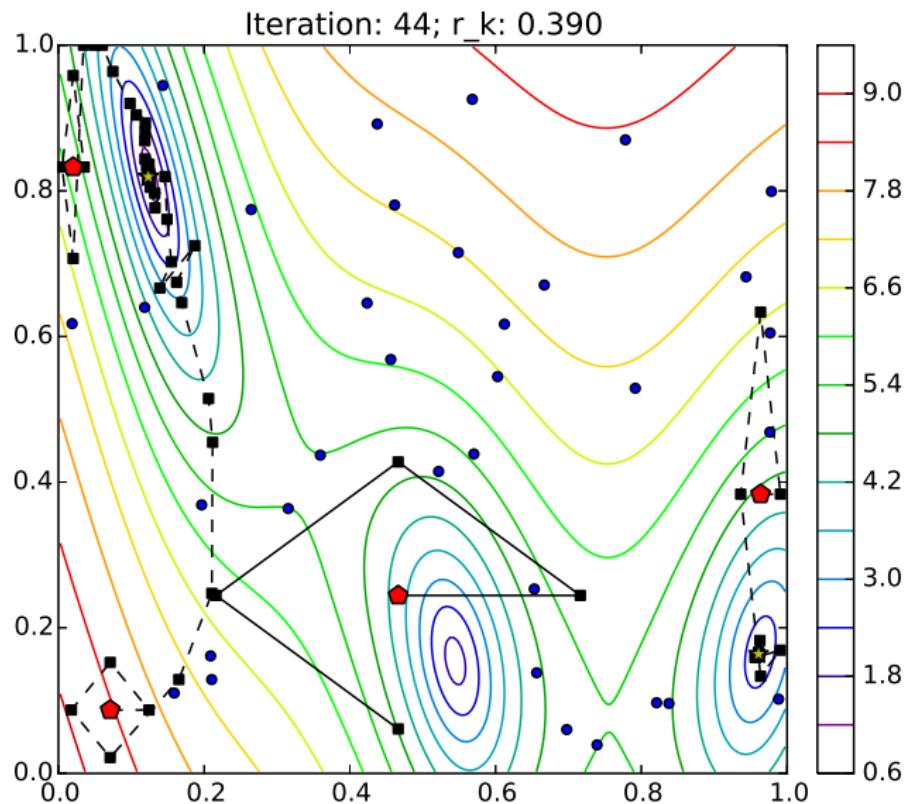


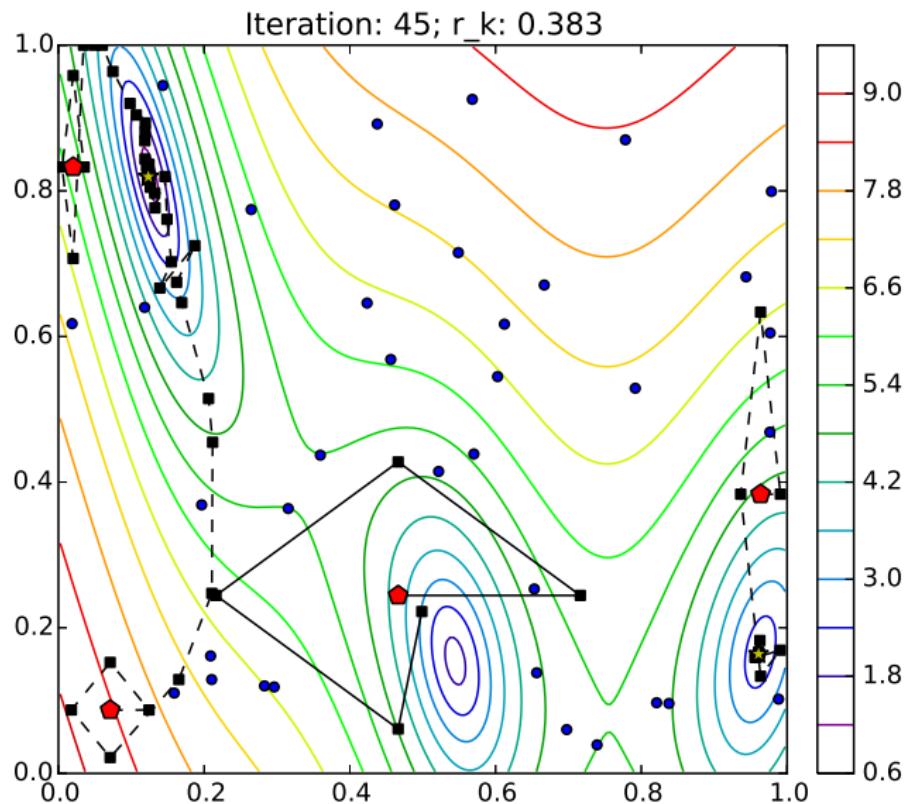


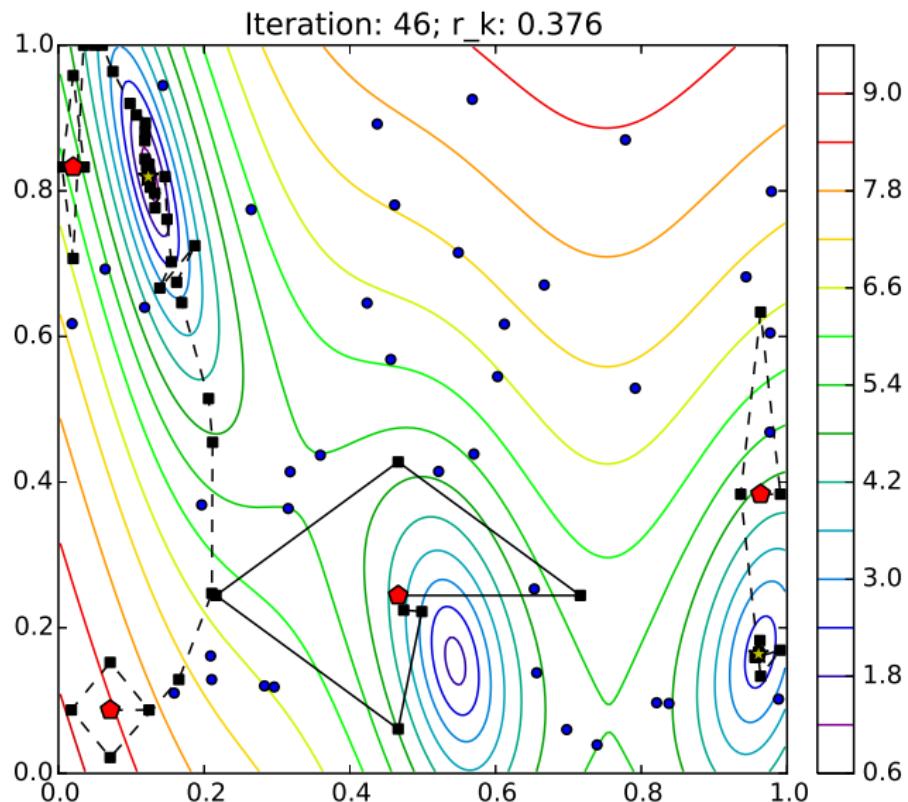


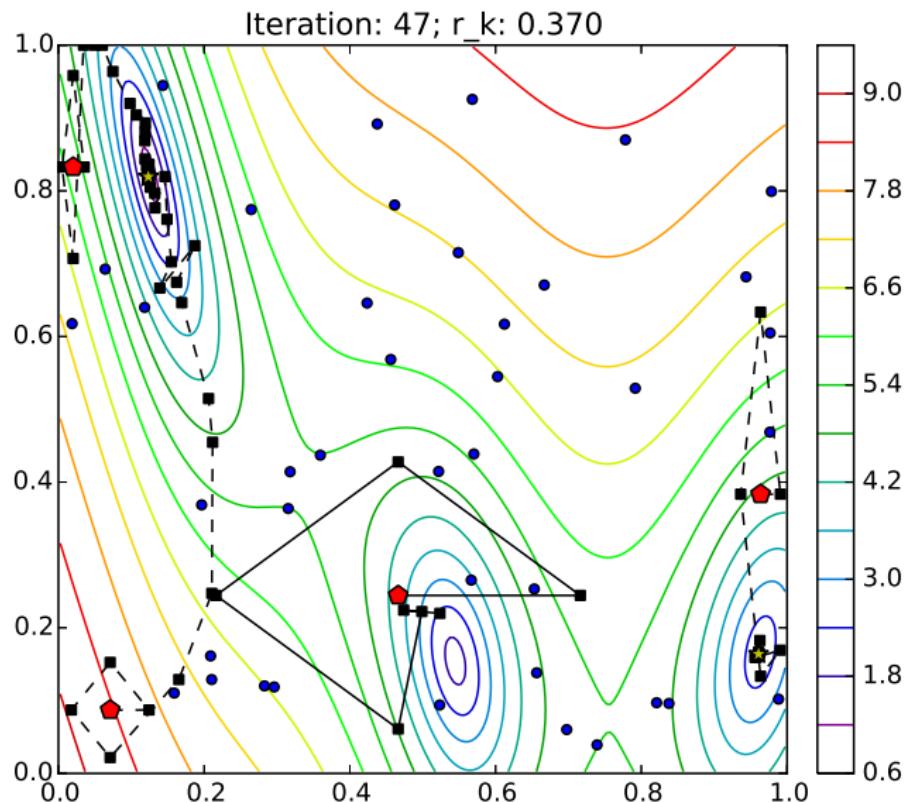


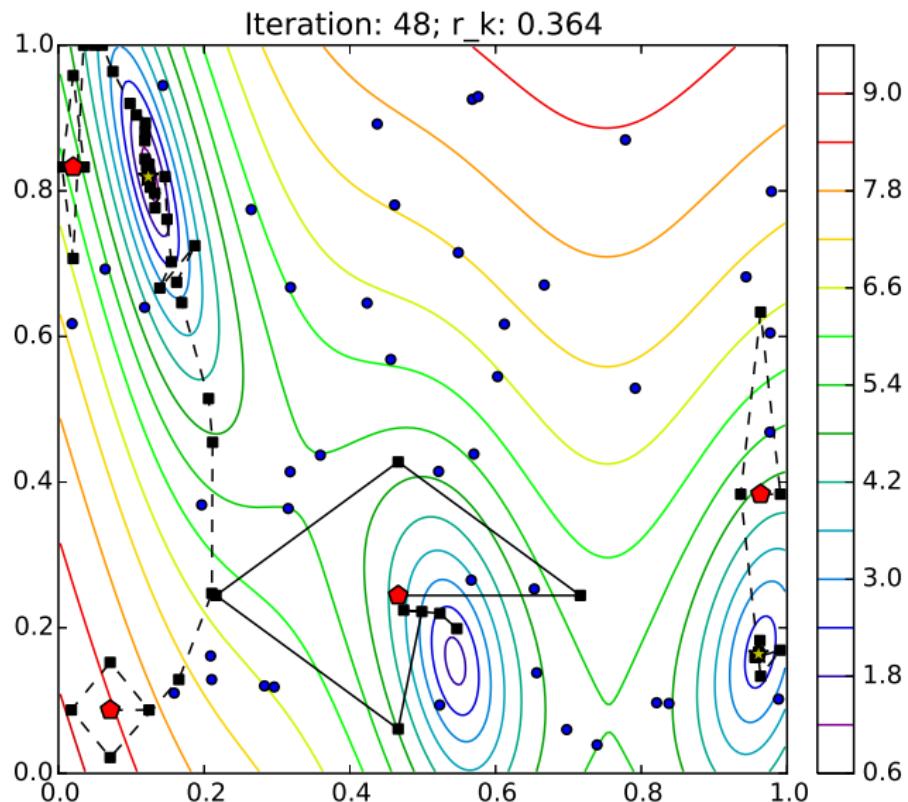


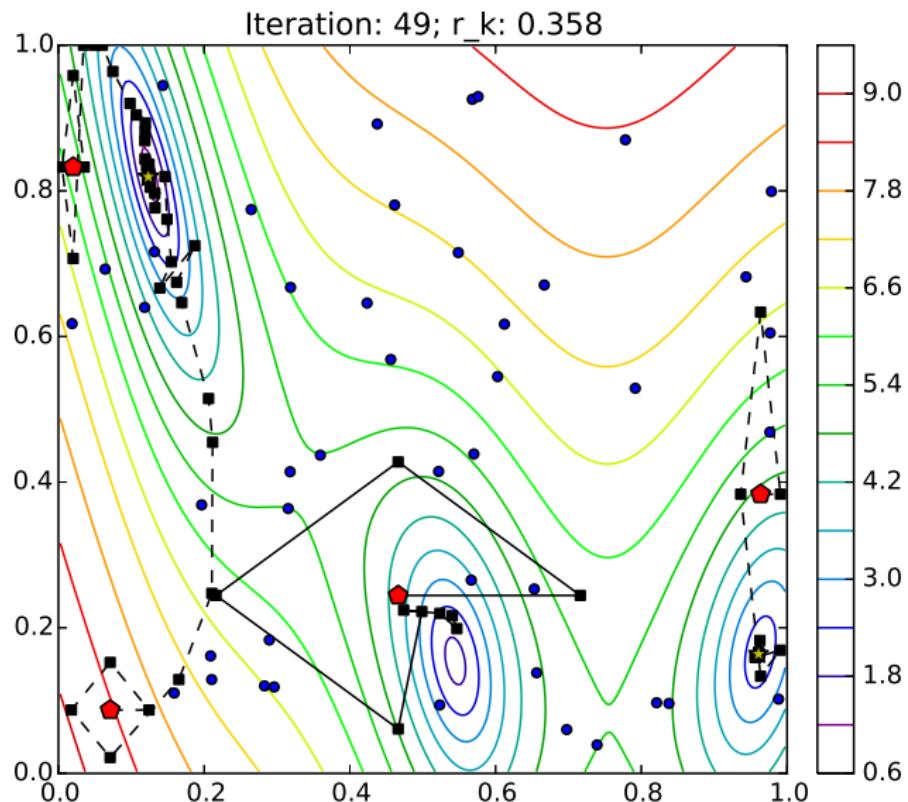


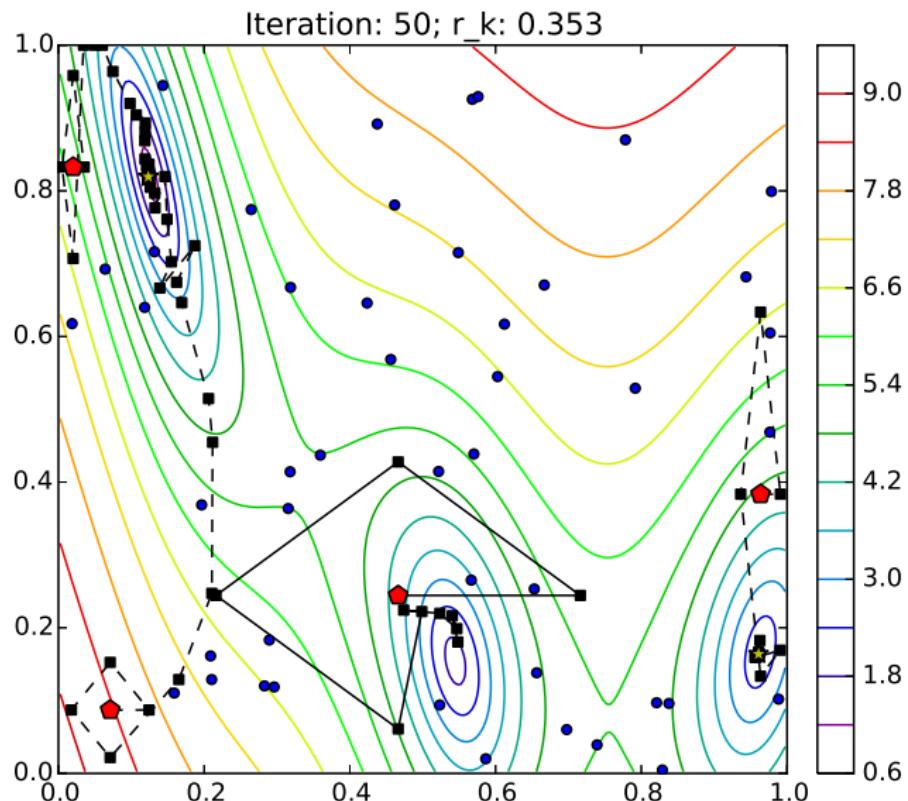












## Desirable properties

- ▶ Honors a starting point
- ▶ Honors bound constraints



## Desirable properties

- ▶ Honors a starting point
- ▶ Honors bound constraints

ORBIT satisfies these [Wild, Regis, Shoemaker, SIAM-JOSC, 2008]

BOBYQA satisfies these [Powell, 2009]



## Desirable properties

- ▶ Honors a starting point
- ▶ Honors bound constraints

ORBIT satisfies these [Wild, Regis, Shoemaker, SIAM-JOSC, 2008]

BOBYQA satisfies these [Powell, 2009]

- ▶ Can return multiple points of interest
- ▶ Reports solution quality/confidence at every iteration
- ▶ Can avoid certain regions in the domain
- ▶ Benefits from additional points mid-run
- ▶ Benefits from a history of past evaluations of  $f$



# Measuring Performance

GLODS Global & local optimization using direct search [Custódio, Madeira (JOGO, 2014)]

Direct Serial DIRECT [D. Finkel's MATLAB code]

pVTDirect Parallel DIRECT [He, Watson, Sosonkina (TOMS, 2009)]

Random Uniform sampling over domain (as a baseline)

BAMLM With local solver ORBIT. Concurrency=4

- ▶ Each method evaluates Direct's  $2n + 1$  initial points.



# Measuring Performance

Notation:

Let  $\mathcal{X}^*$  be the set of all local minima of  $f$ .

Let  $f_{(i)}^*$  be the  $i$ th smallest value  $\{f(x^*) | x^* \in \mathcal{X}^*\}$ .

Let  $x_{(i)}^*$  be the element of  $\mathcal{X}^*$  corresponding to the value  $f_{(i)}^*$ .

The global minimum has been found at a level  $\tau > 0$  at batch  $k$  if an algorithm it has found a point  $\hat{x}$  satisfying:

$$f(\hat{x}) - f_{(1)}^* \leq (1 - \tau) \left( f(x_0) - f_{(1)}^* \right),$$

where  $x_0$  is the starting point for problem  $p$ .



# Measuring Performance

Notation:

Let  $\mathcal{X}^*$  be the set of all local minima of  $f$ .

Let  $f_{(i)}^*$  be the  $i$ th smallest value  $\{f(x^*) | x^* \in \mathcal{X}^*\}$ .

Let  $x_{(i)}^*$  be the element of  $\mathcal{X}^*$  corresponding to the value  $f_{(i)}^*$ .

The  $j$  best local minima have been found at a level  $\tau > 0$  at batch  $k$  if:

$$\left| \left\{ x_{(1)}^*, \dots, x_{(\underline{j}-1)}^* \right\} \cap \left\{ x_{(i)}^* : \exists x \in \mathcal{H}_k \text{ with } \|x - x_{(i)}^*\| \leq r_n(\tau) \right\} \right| = \underline{j} - 1$$

&

$$\left| \left\{ x_{(\underline{j})}^*, \dots, x_{(\bar{j})}^* \right\} \cap \left\{ x_{(i)}^* : \exists x \in \mathcal{H}_k \text{ with } \|x - x_{(i)}^*\| \leq r_n(\tau) \right\} \right| \geq j - \underline{j} + 1,$$

where  $r_n(\tau) = \sqrt[n]{\frac{\tau \operatorname{vol}(\mathcal{D}) \Gamma(\frac{n}{2} + 1)}{\pi^{n/2}}}$ . where  $\bar{j}$  and  $\underline{j}$  are the largest and smallest integers such that  $f_{(\bar{j})}^* = f_{(\underline{j})}^* = f_{(\underline{j})}^*$ .

## Problems considered

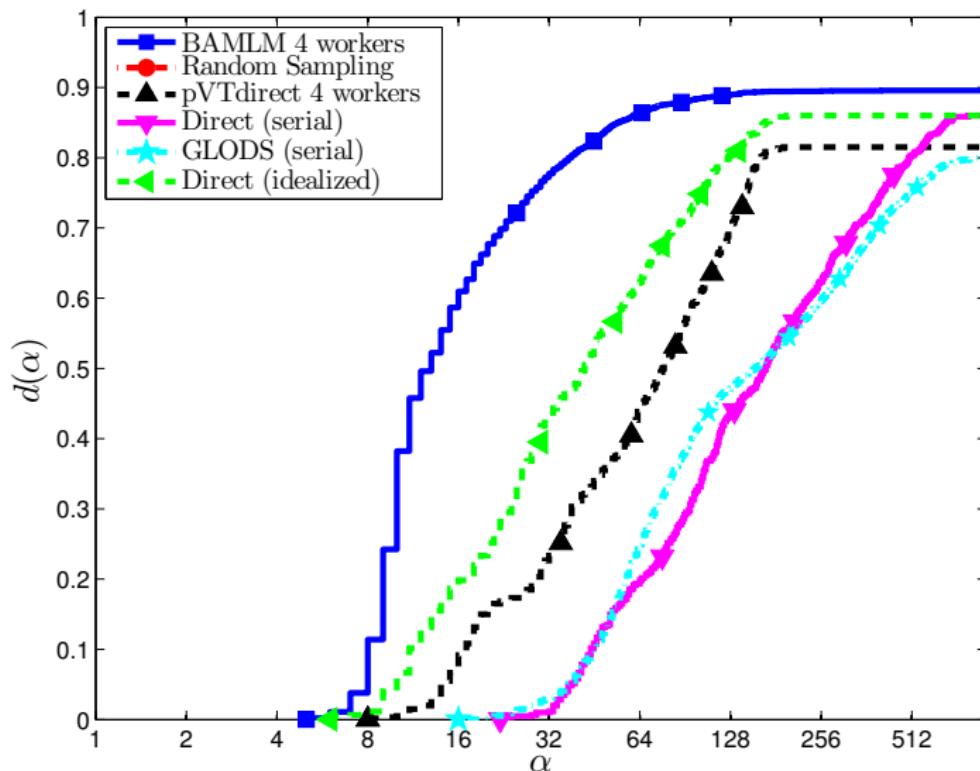
GKLS problem generator [Gaviano et al., "Algorithm 829" (TOMS, 2003)]

- ▶ 600 synthetic problems with known local minima
- ▶  $n = 2, \dots, 7$
- ▶ 10 local minima in the unit cube with a unique global minimum
- ▶ 100 problems for each dimension
- ▶ 5 replications (different seeds) for each problem
- ▶ 5000 evaluations



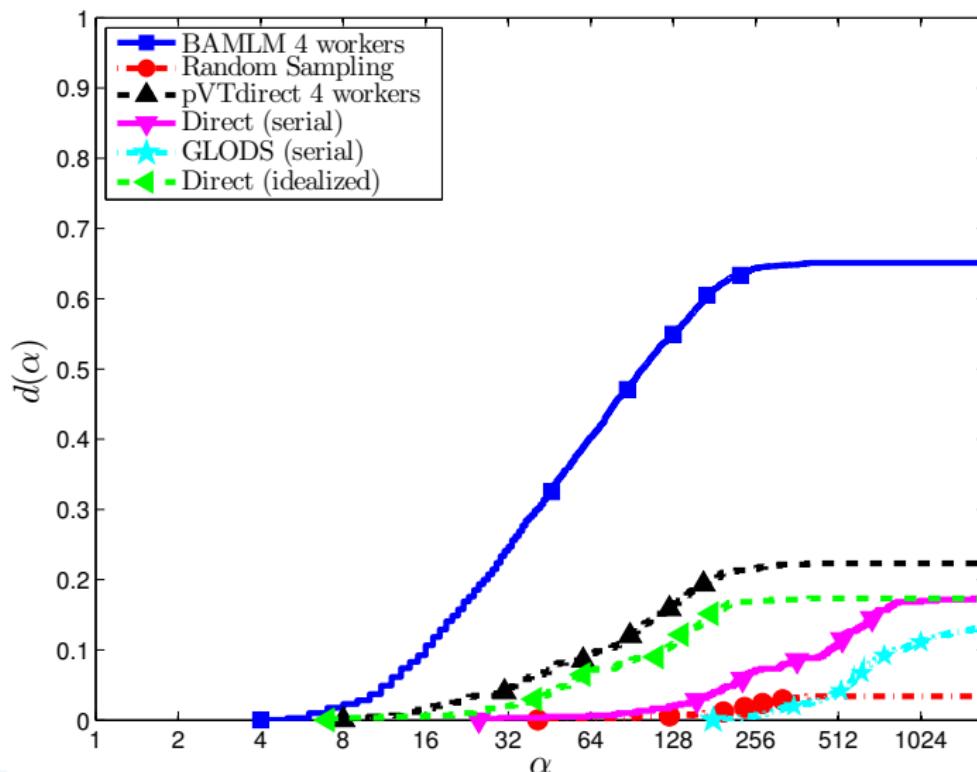
# Data Profiles

$$f(x) - f_{(1)} \leq (1 - 10^{-5}) (f(x_0) - f_{(1)})$$



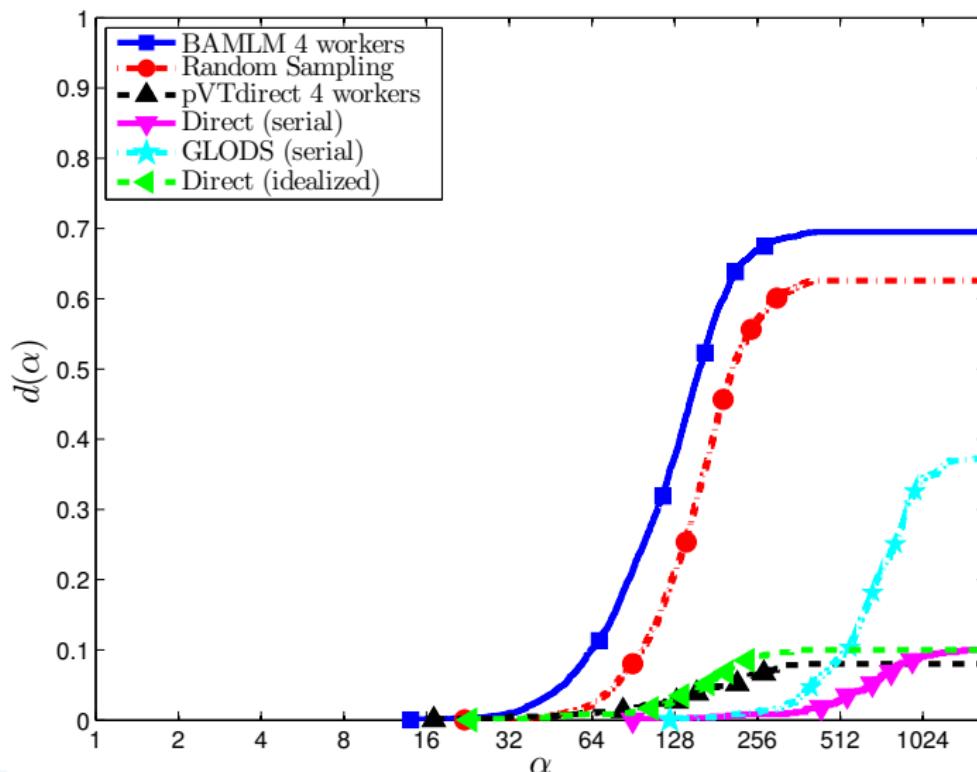
# Data Profiles

Within  $\sqrt{\frac{10^{-4}\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$  of 3 best minima



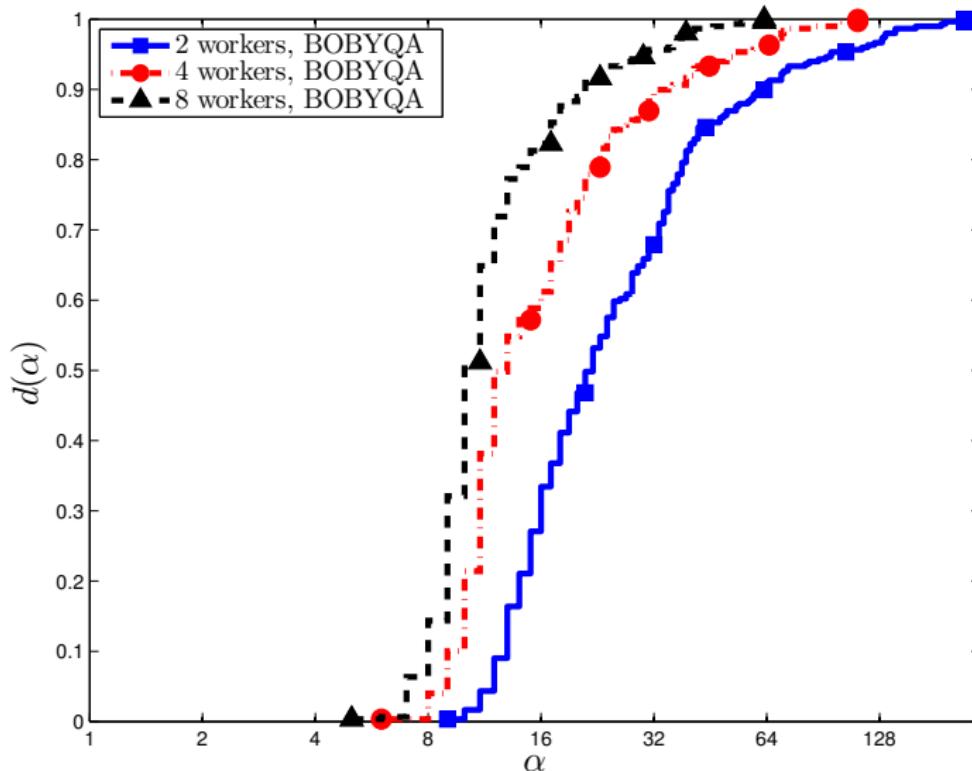
# Data Profiles

Within  $\sqrt{\frac{10^{-3}\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$  of 7 best minima



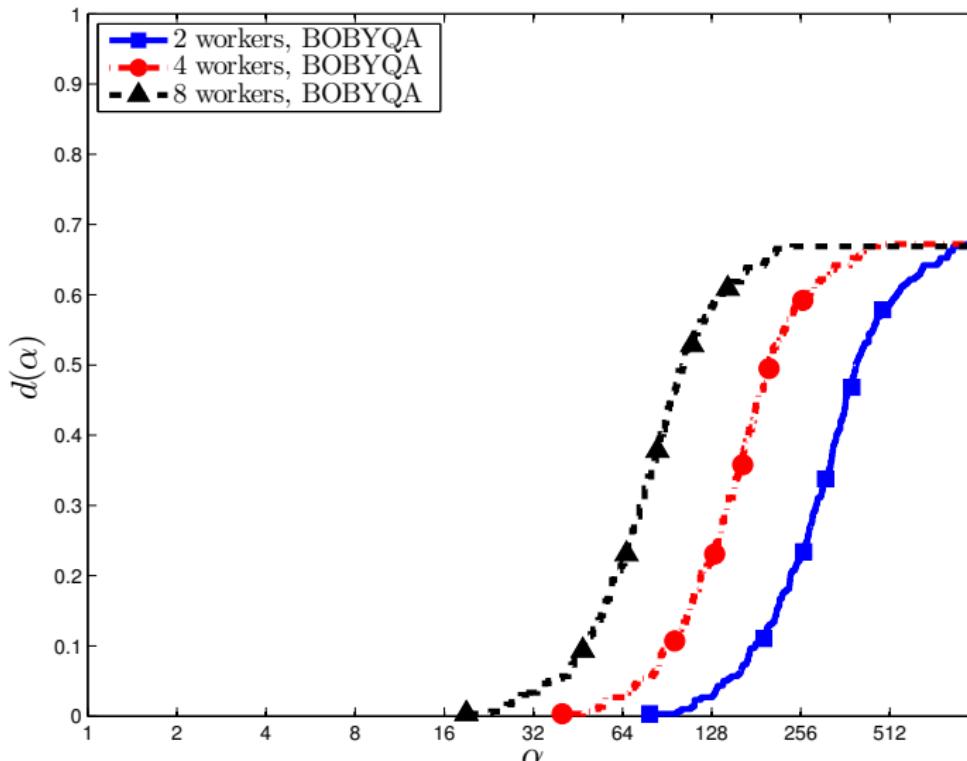
# Data Profiles

$$f(x) - f_{(1)} \leq (1 - 10^{-5}) (f(x_0) - f_{(1)})$$

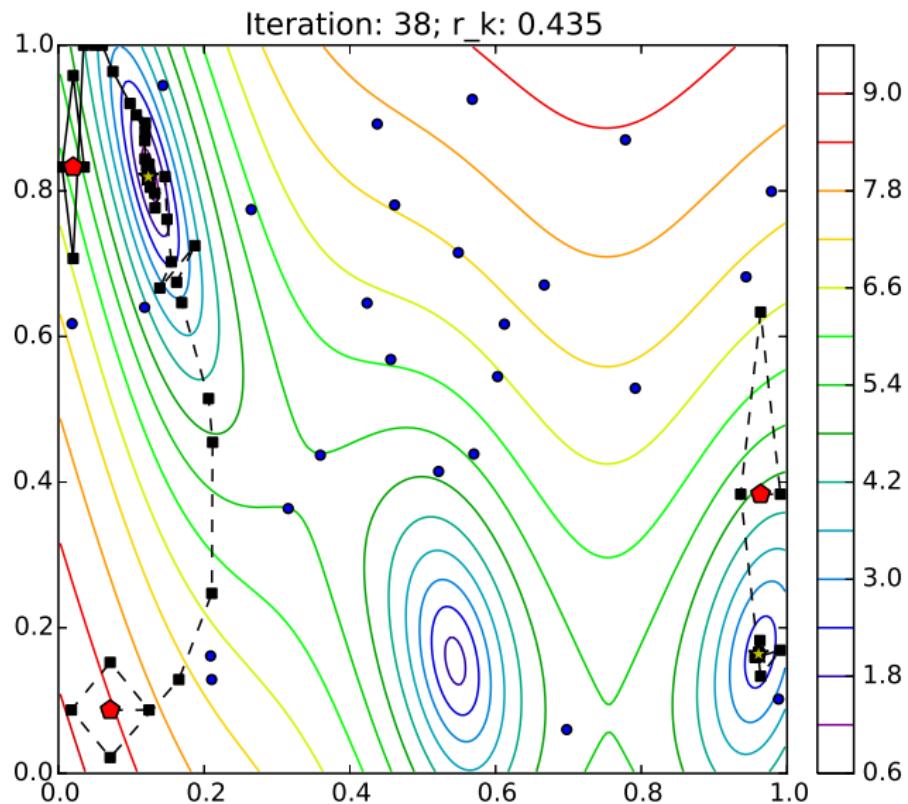


# Data Profiles

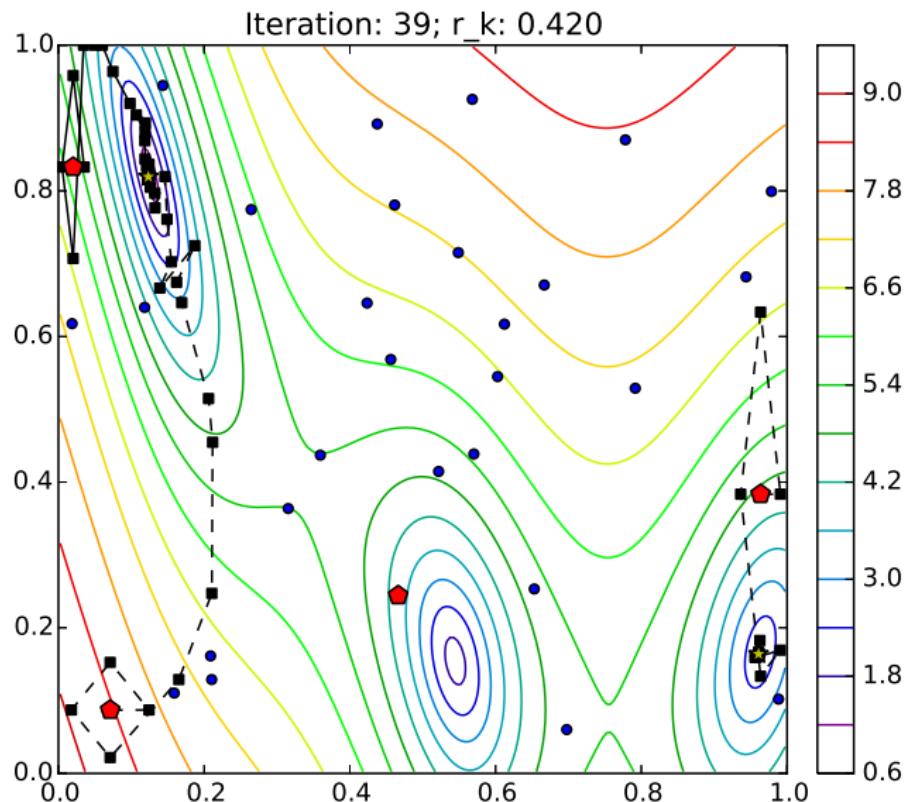
Within  $\sqrt{\frac{10^{-3}\Gamma(\frac{n}{2}+1)}{\pi^{n/2}}}$  of 7 best minima



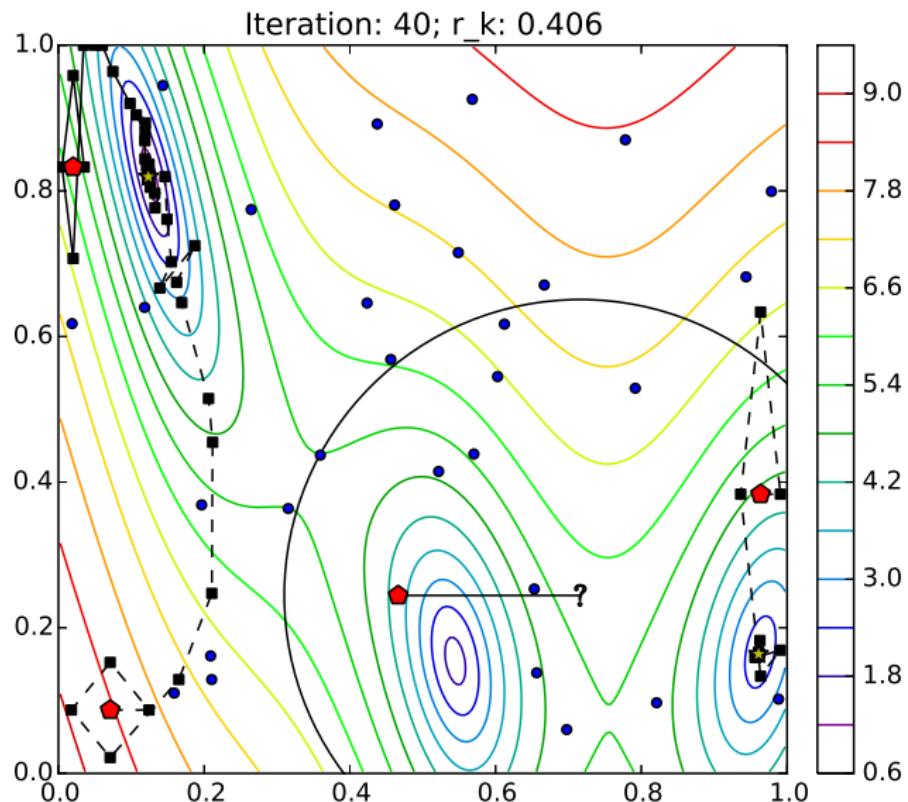
## Pausing runs



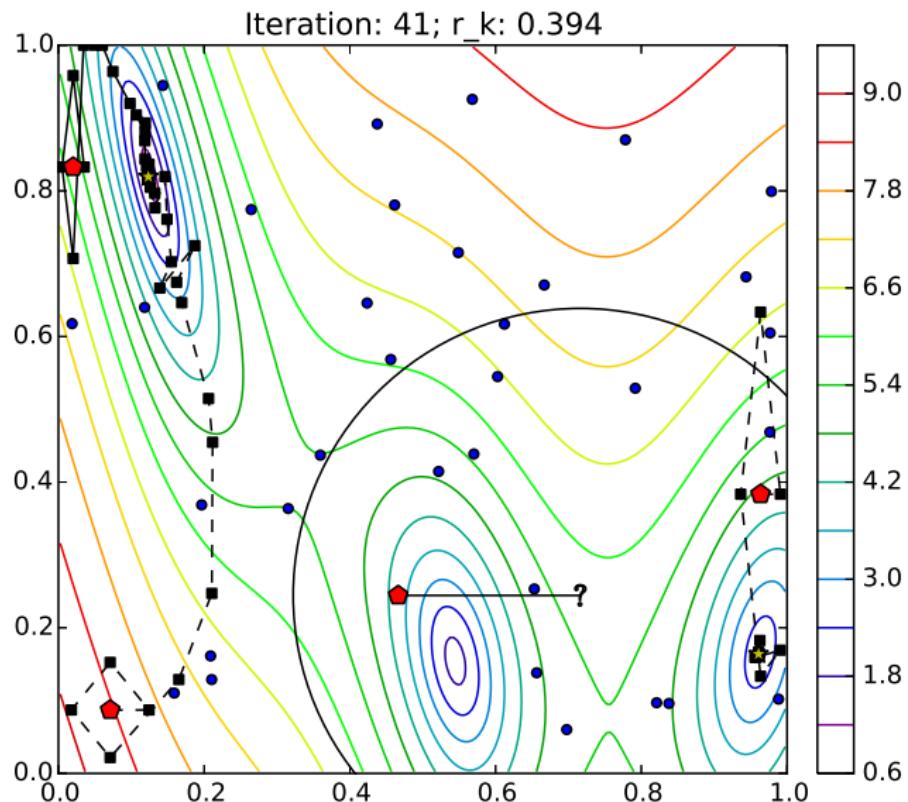
## Pausing runs



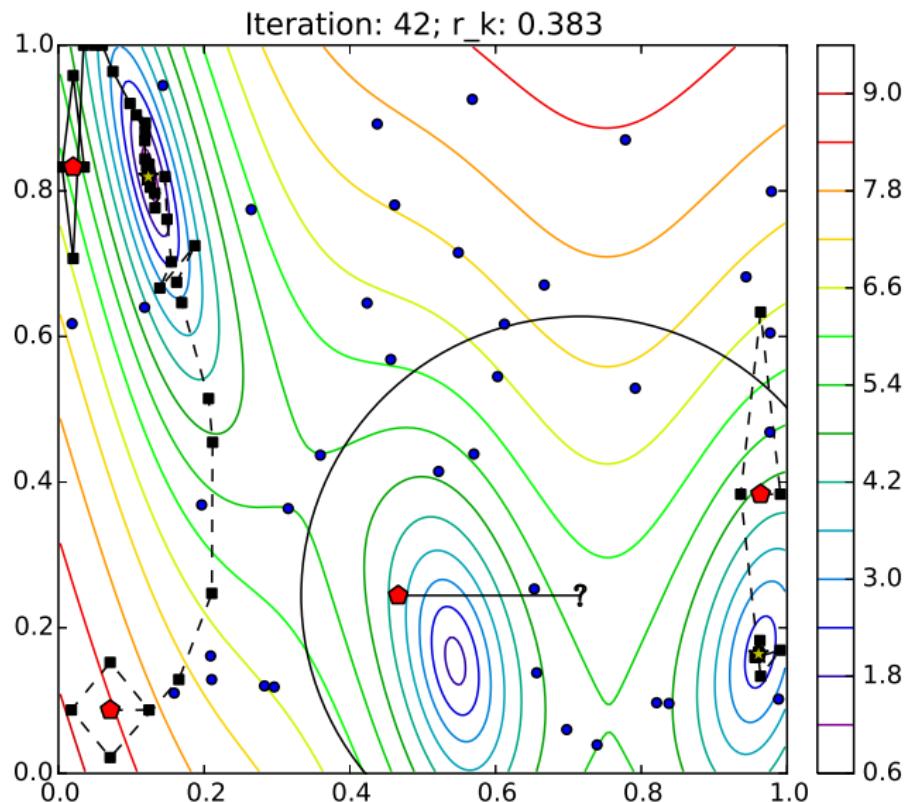
## Pausing runs



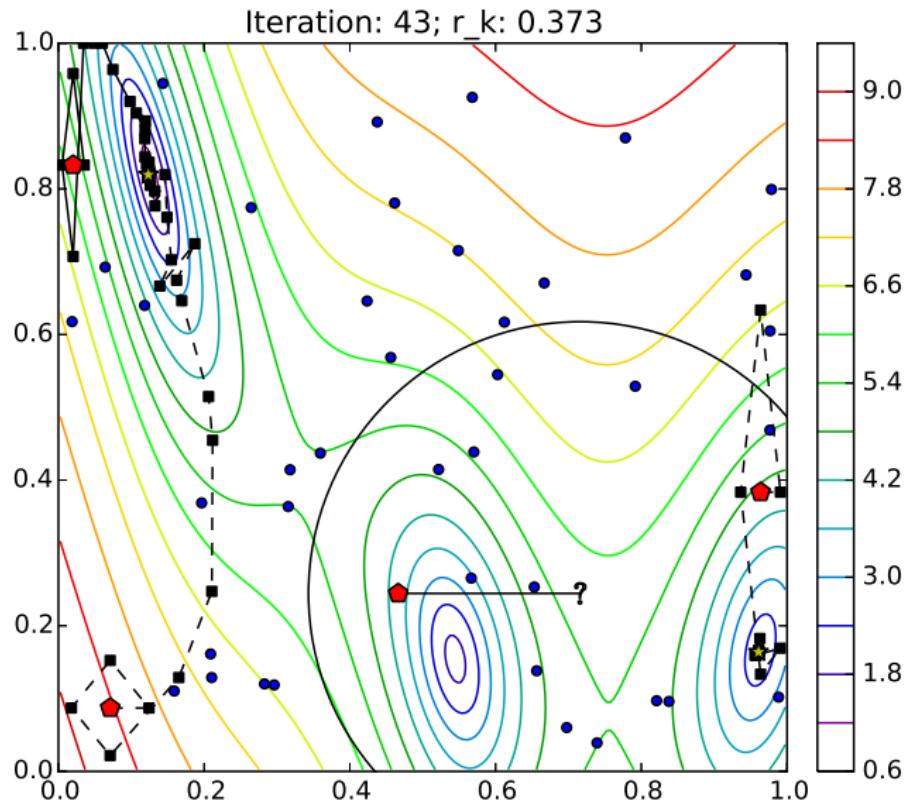
## Pausing runs



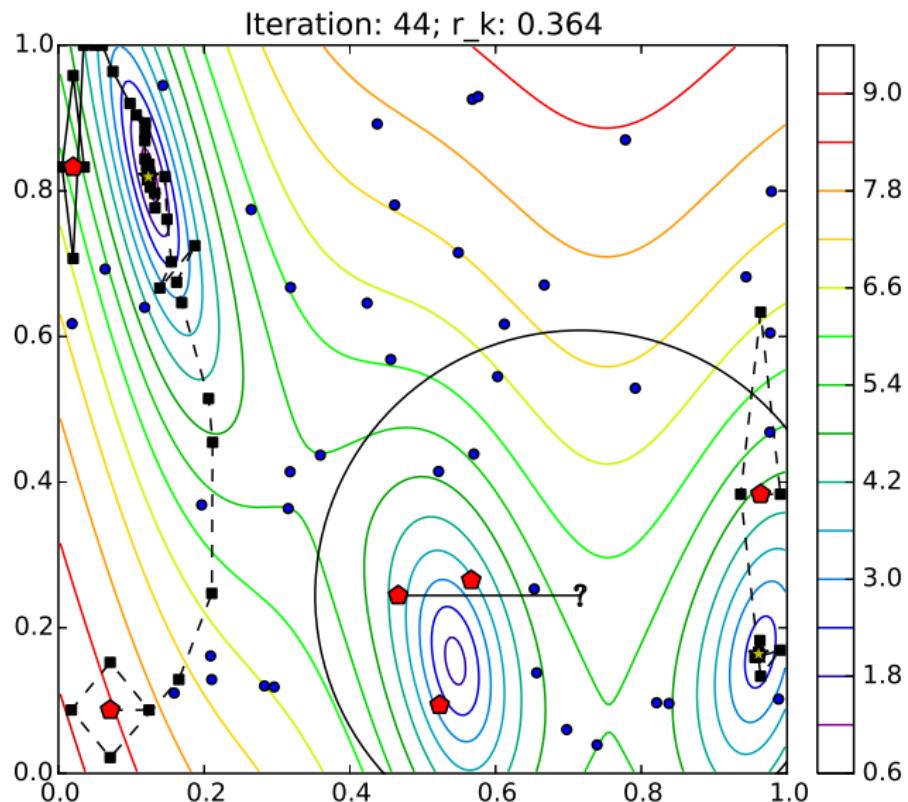
## Pausing runs



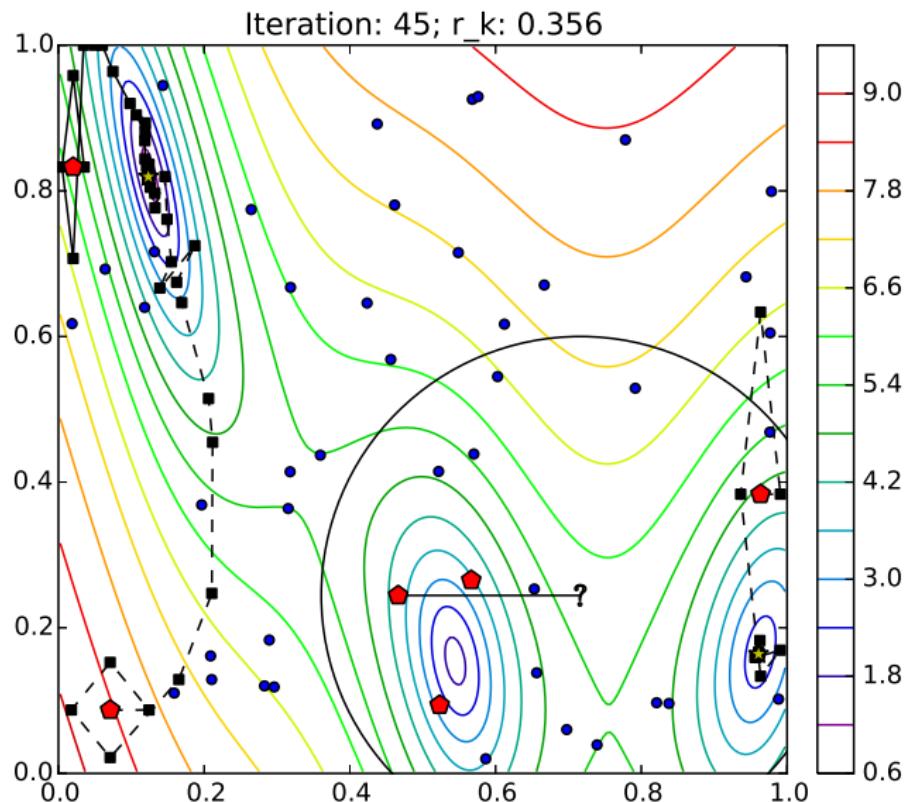
## Pausing runs



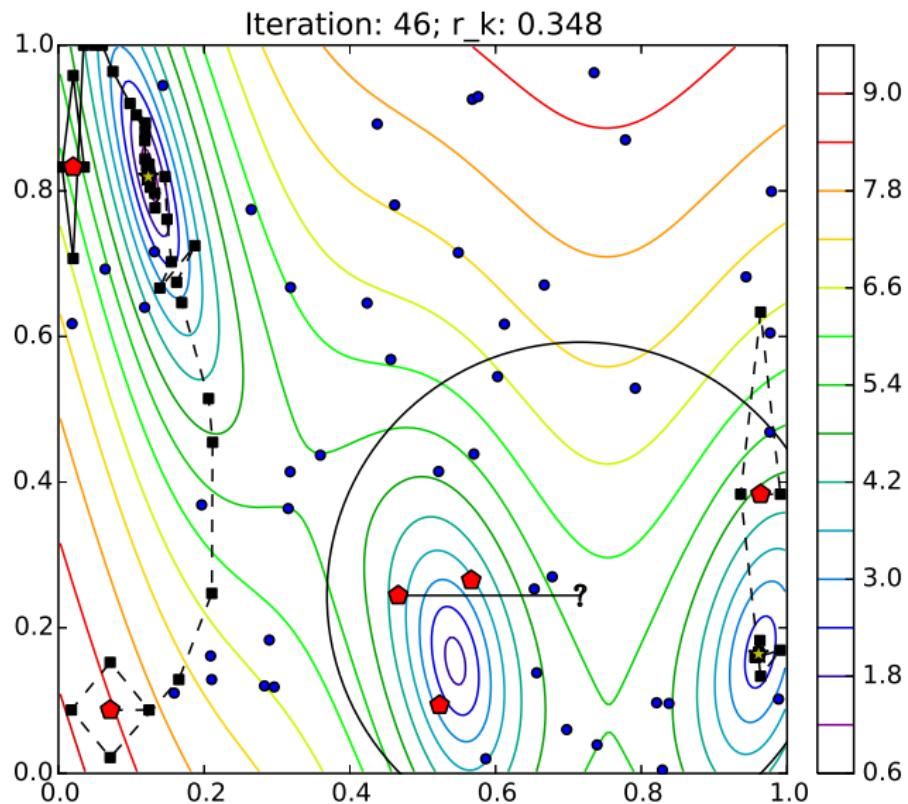
## Pausing runs



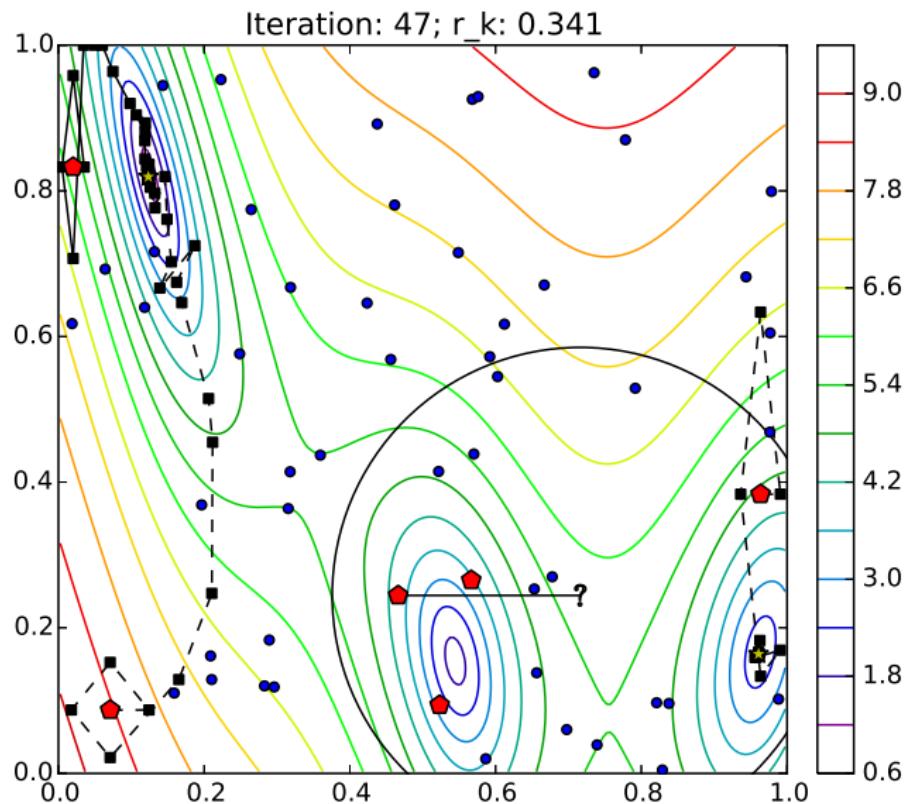
## Pausing runs



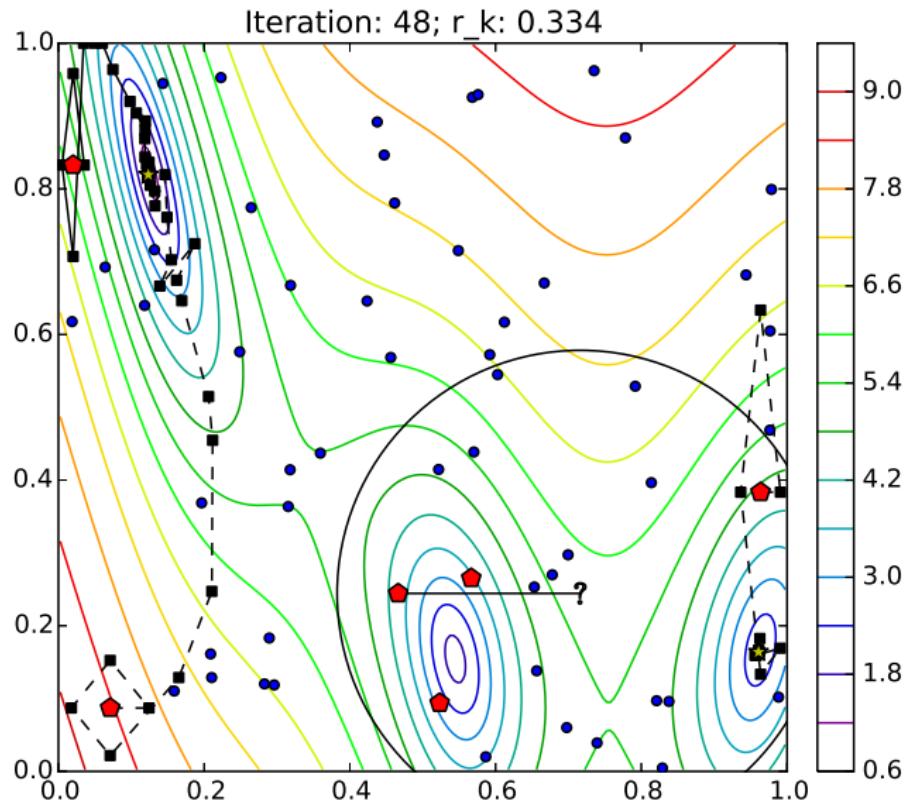
## Pausing runs



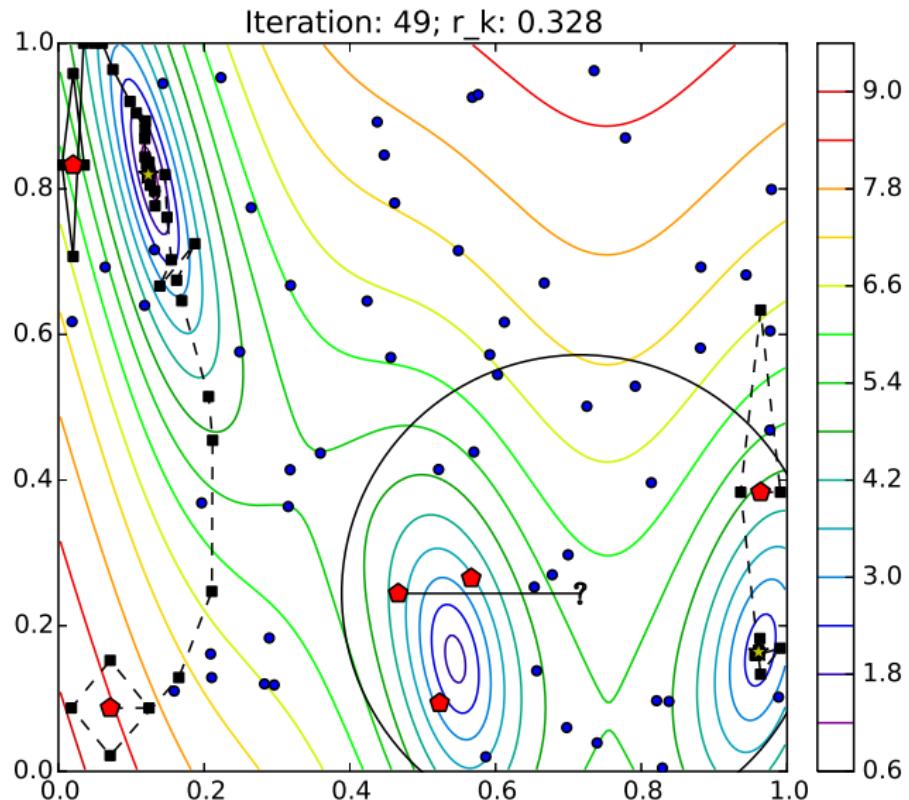
## Pausing runs



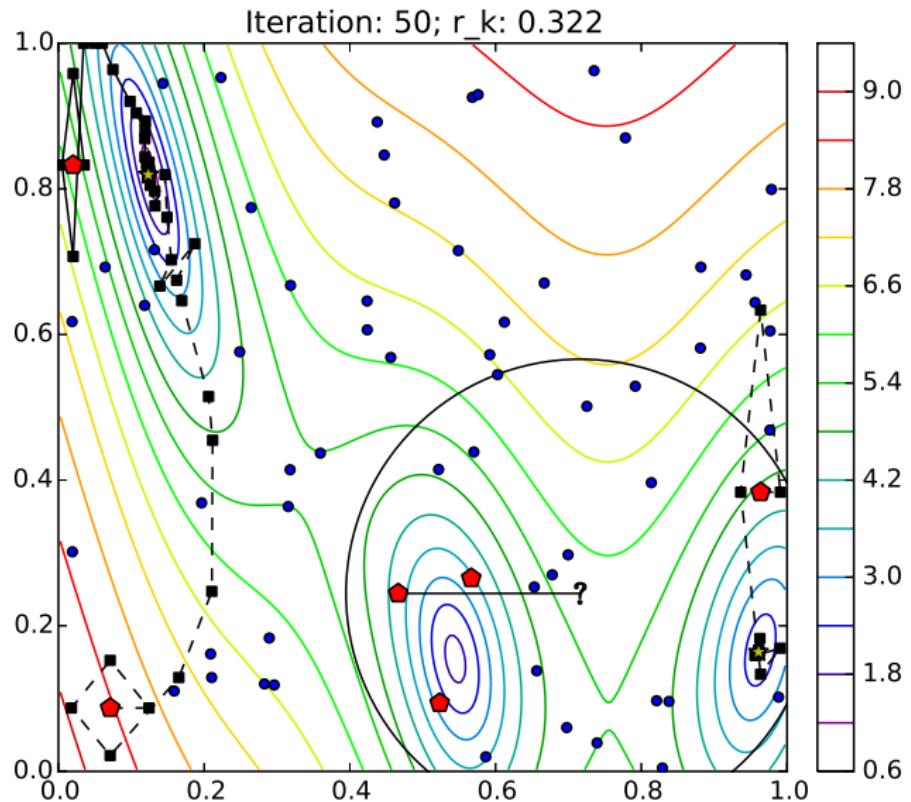
## Pausing runs



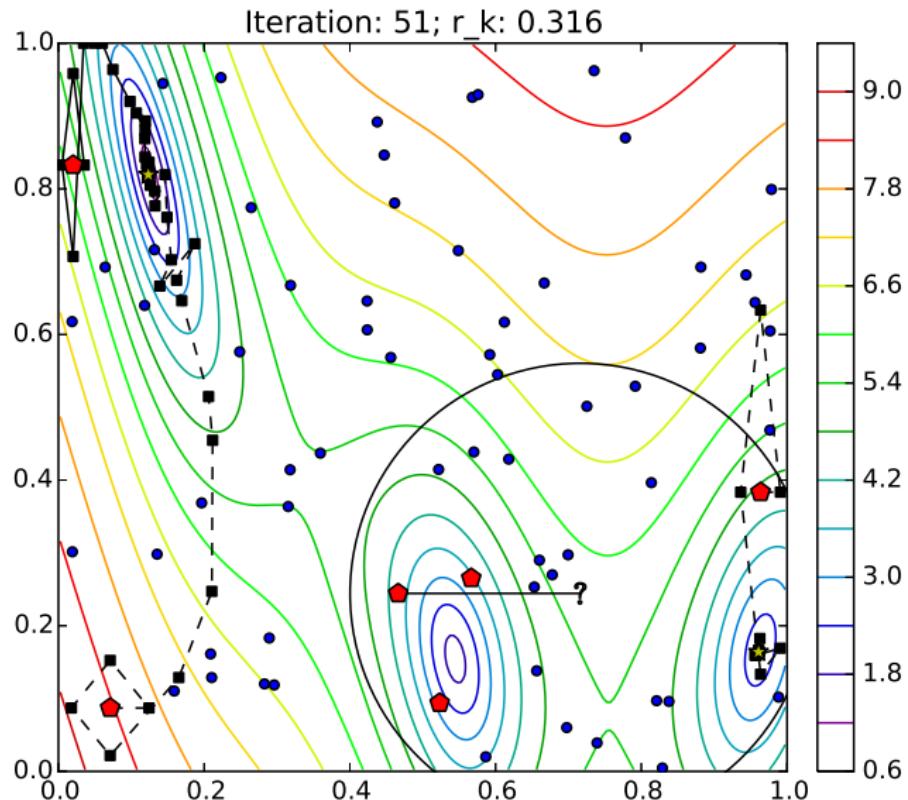
## Pausing runs



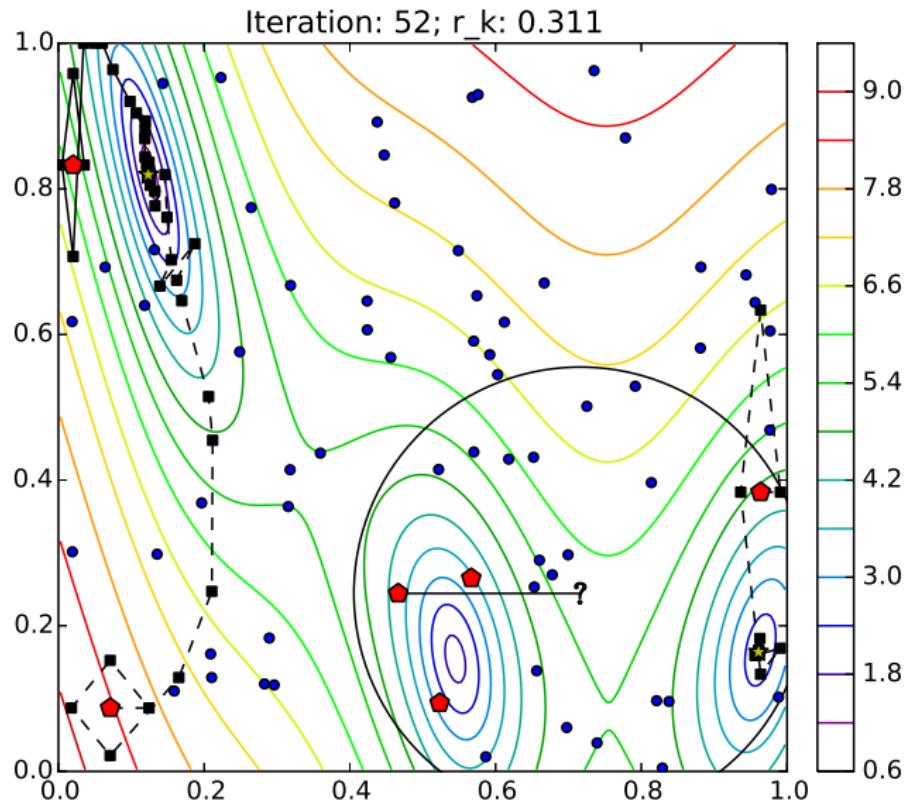
## Pausing runs



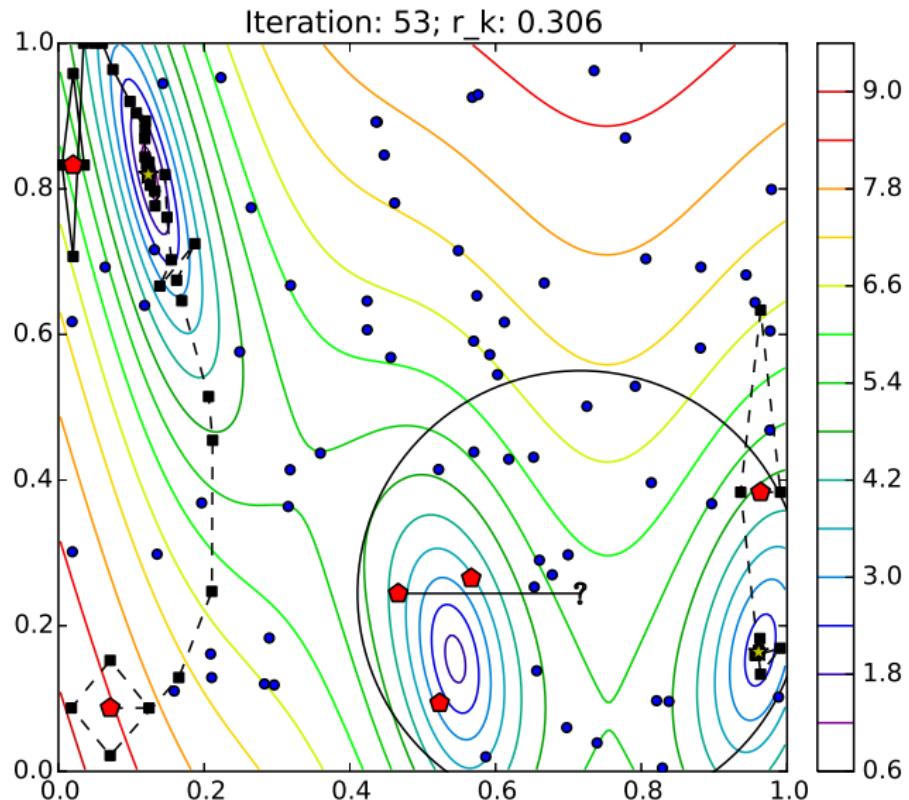
## Pausing runs



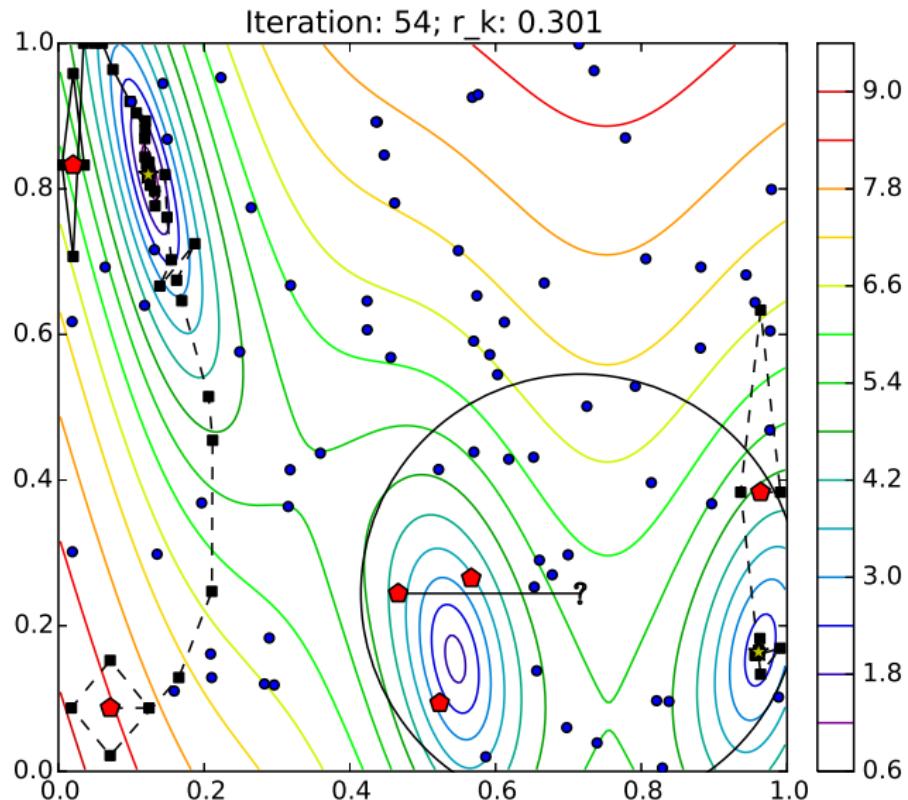
## Pausing runs



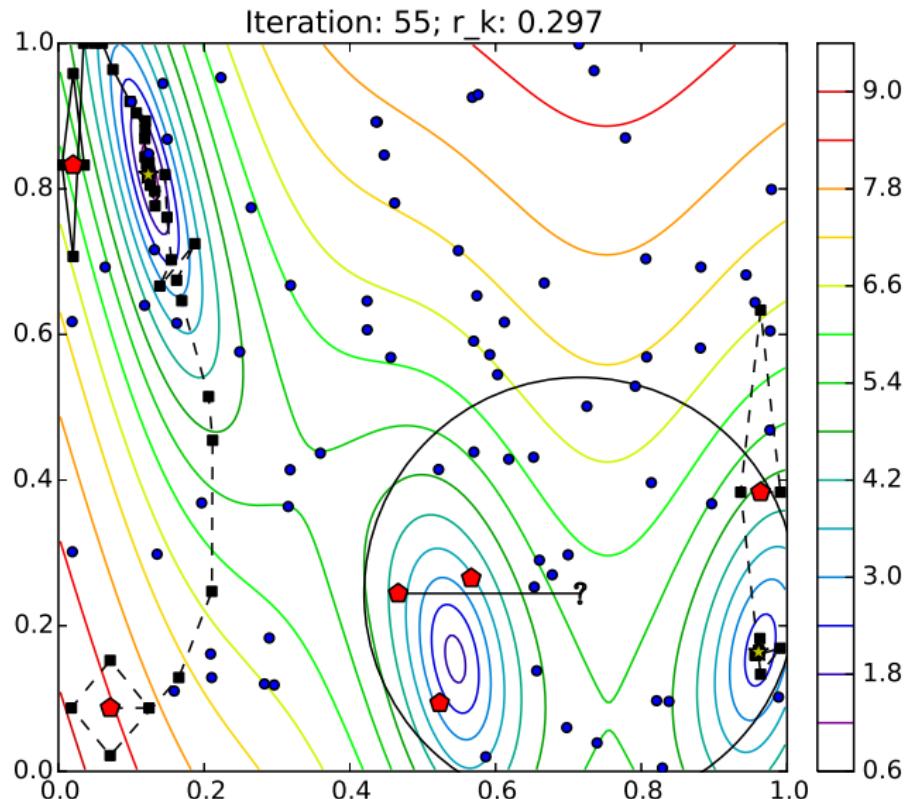
## Pausing runs



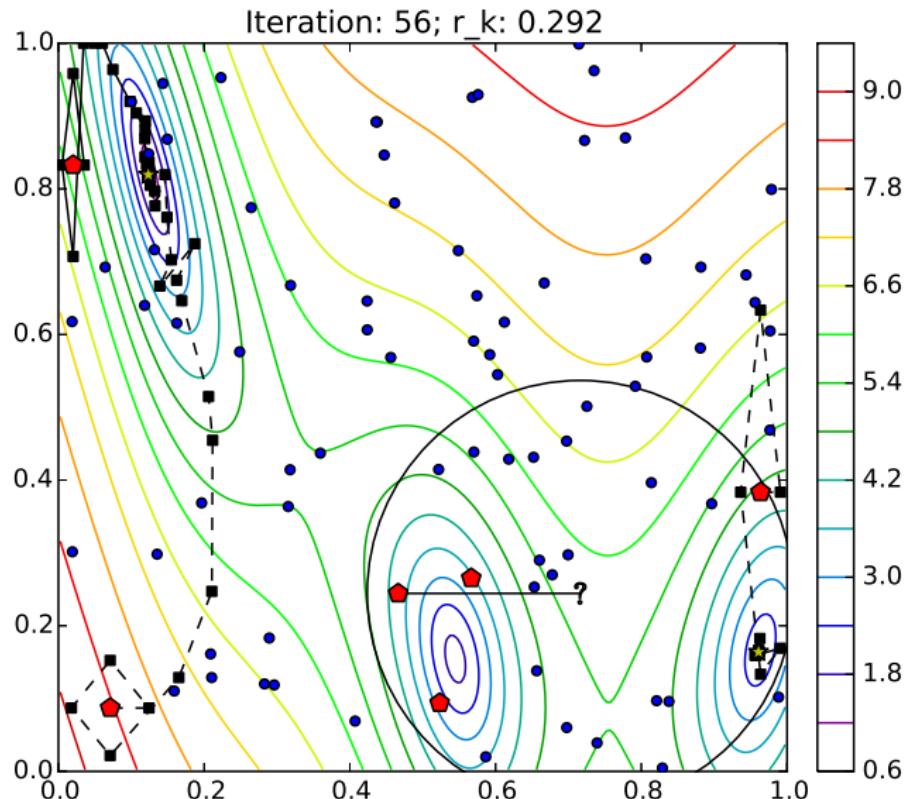
## Pausing runs



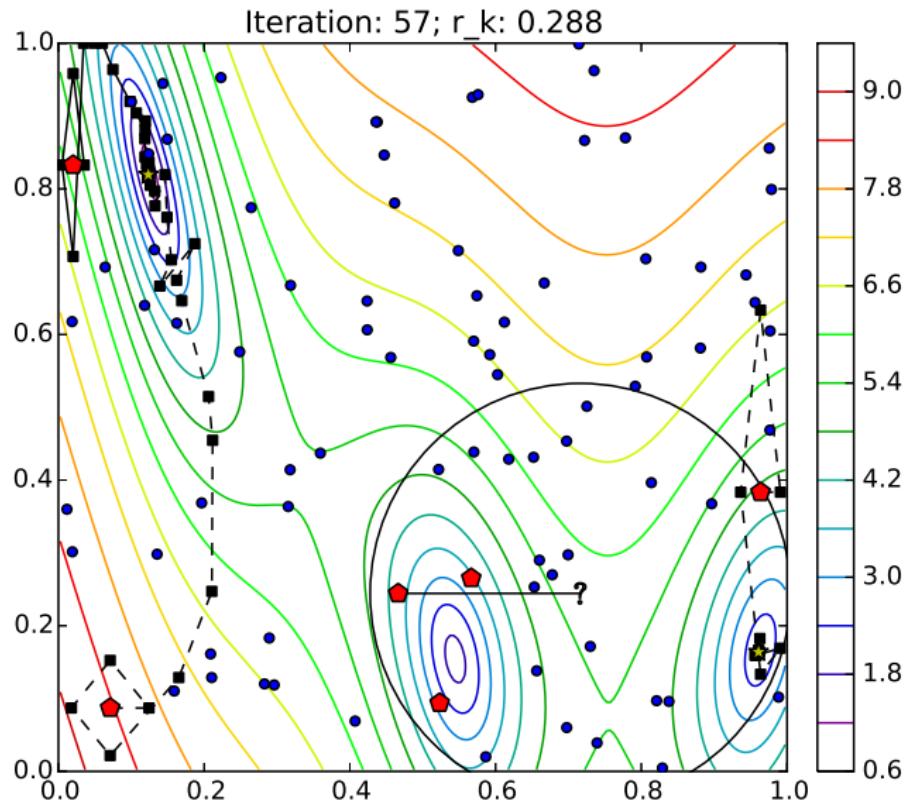
## Pausing runs



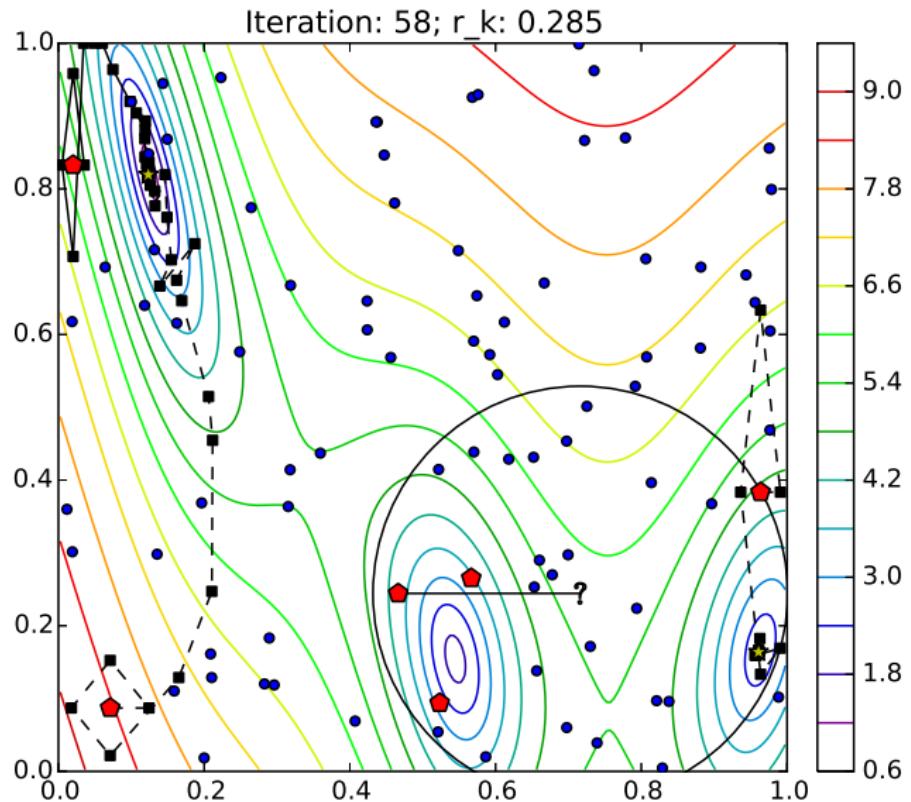
## Pausing runs



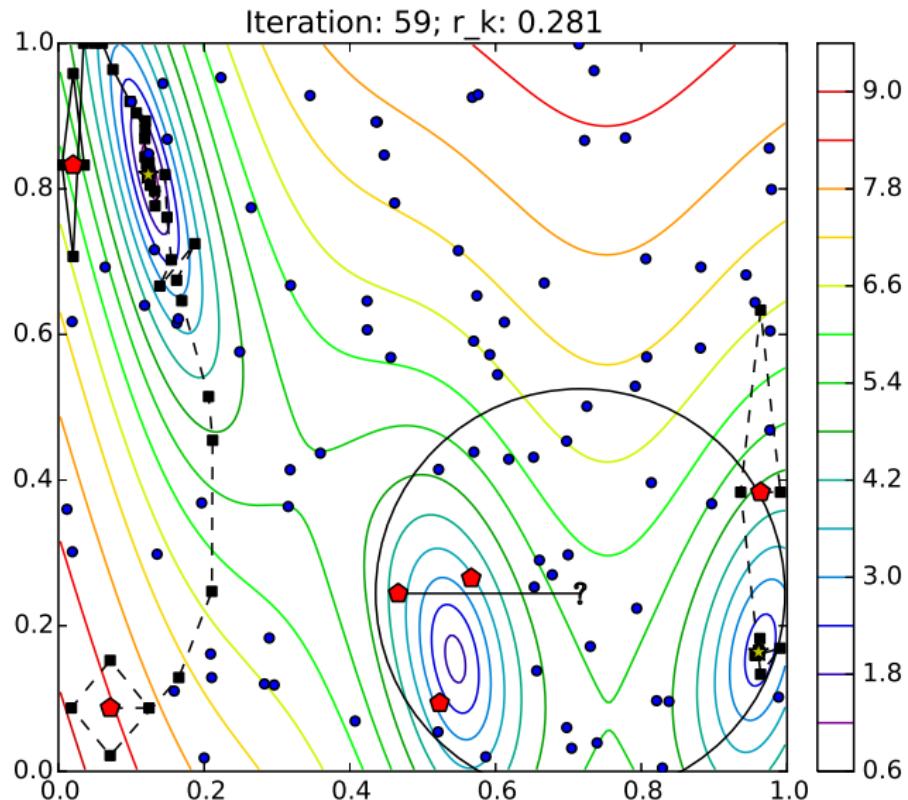
## Pausing runs



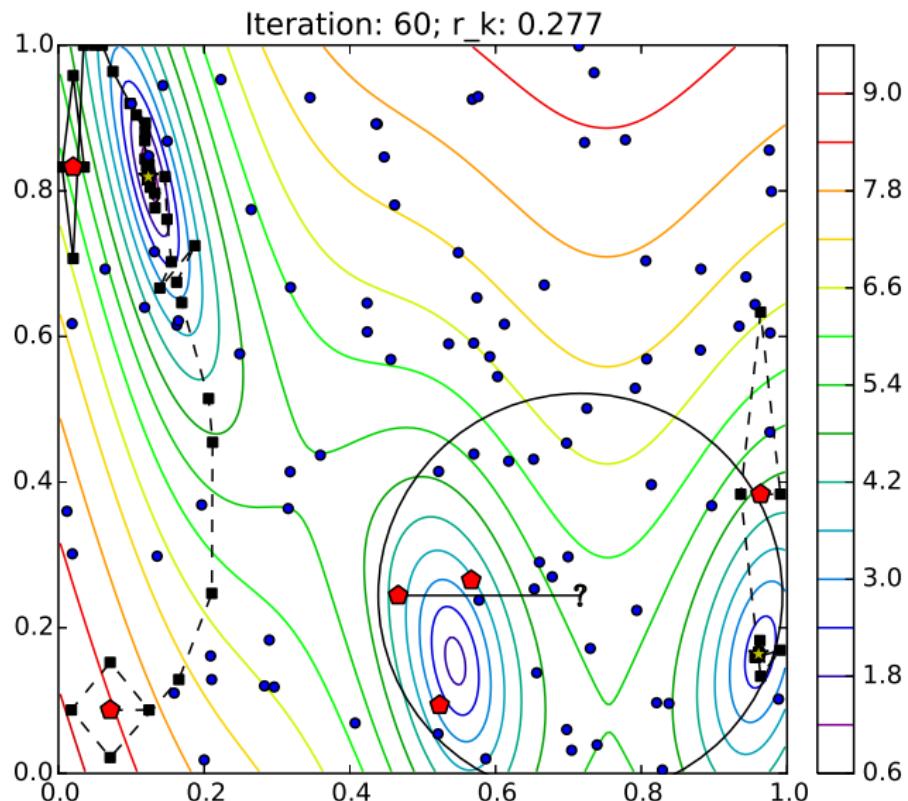
## Pausing runs



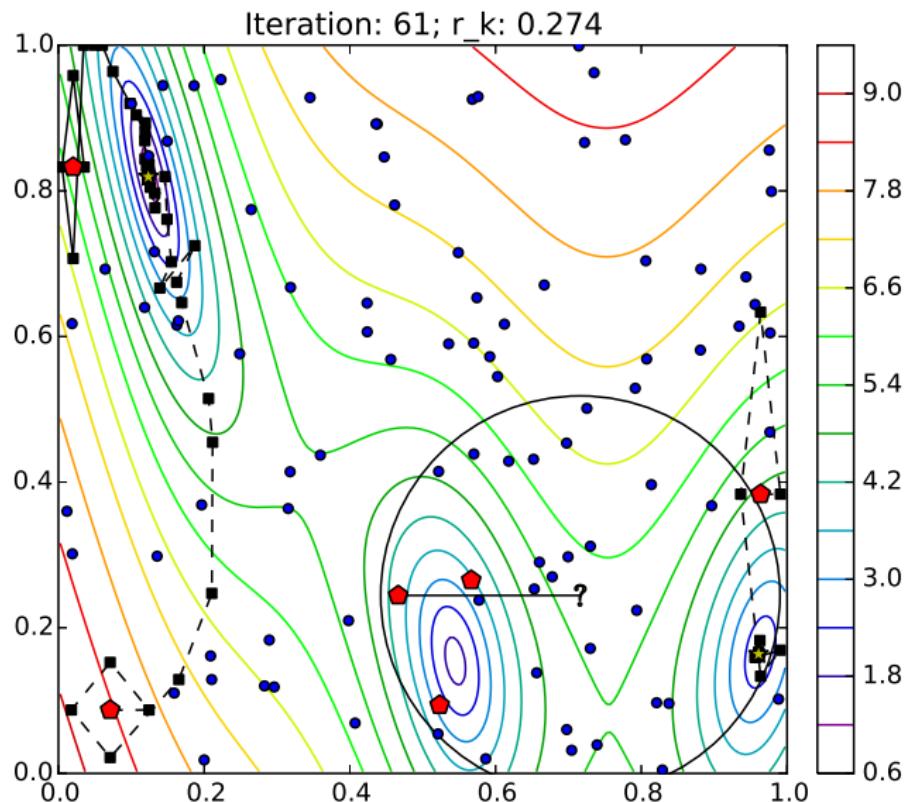
## Pausing runs



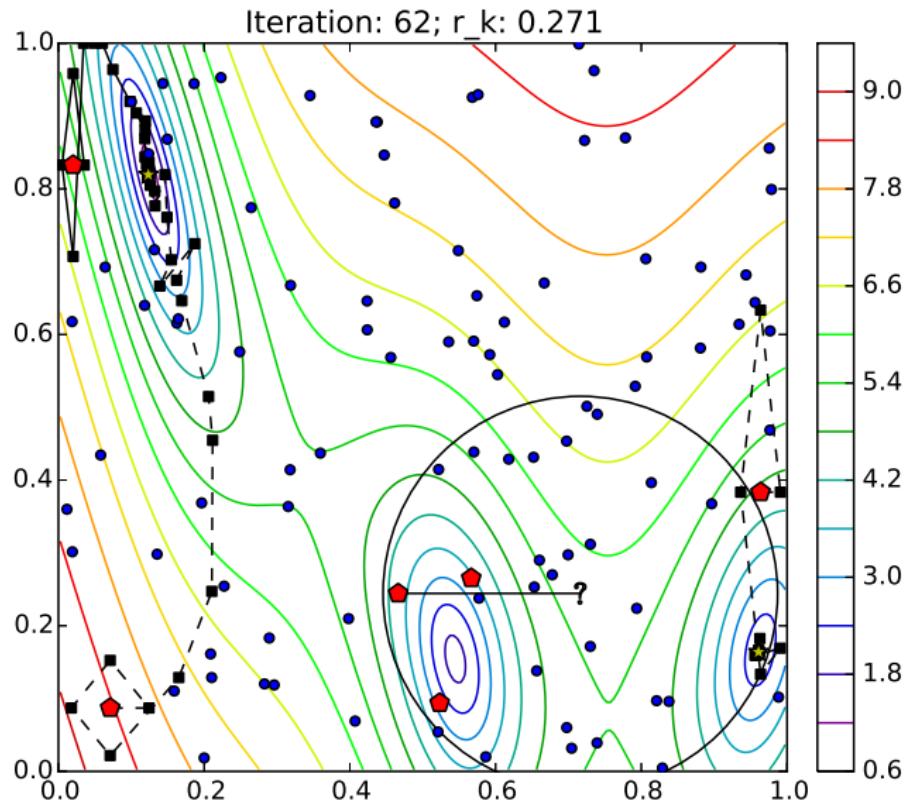
## Pausing runs



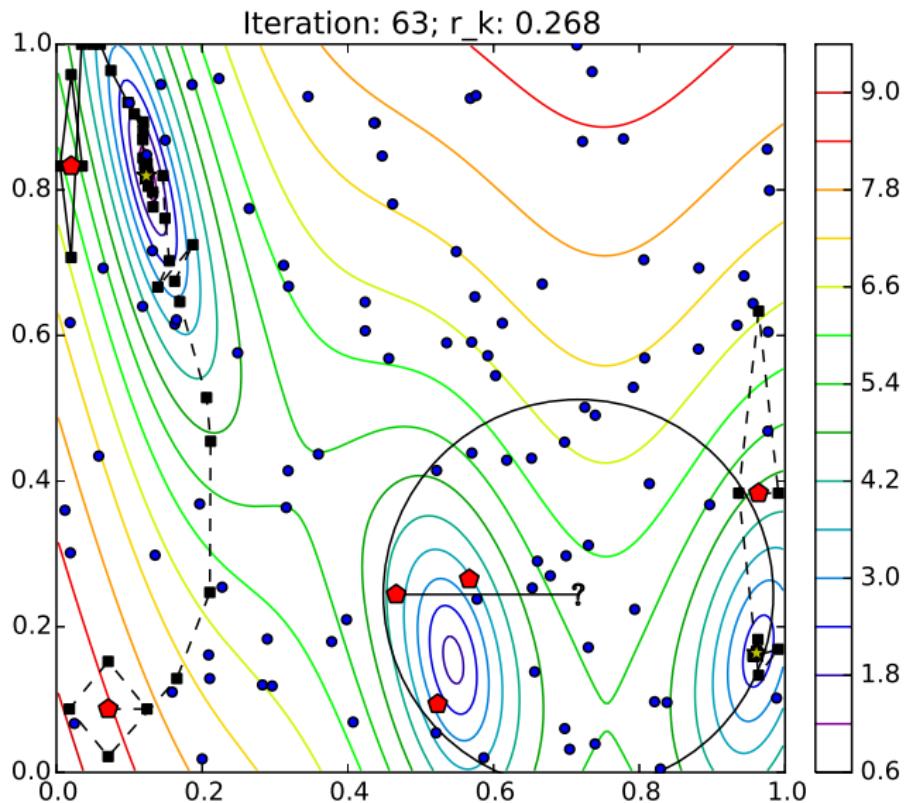
## Pausing runs



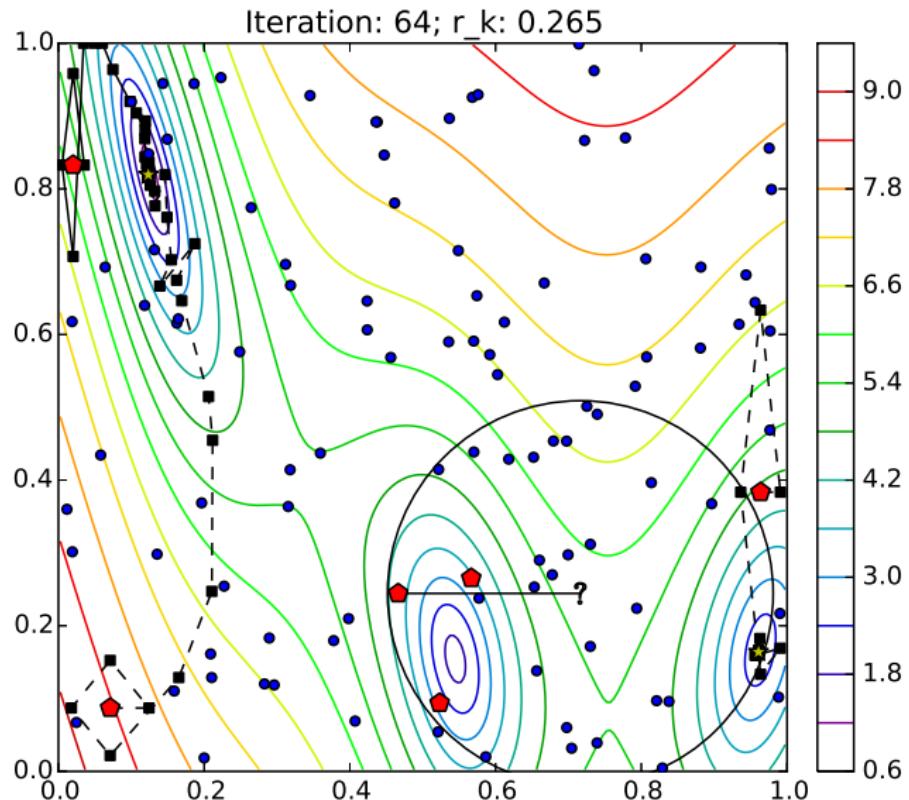
## Pausing runs



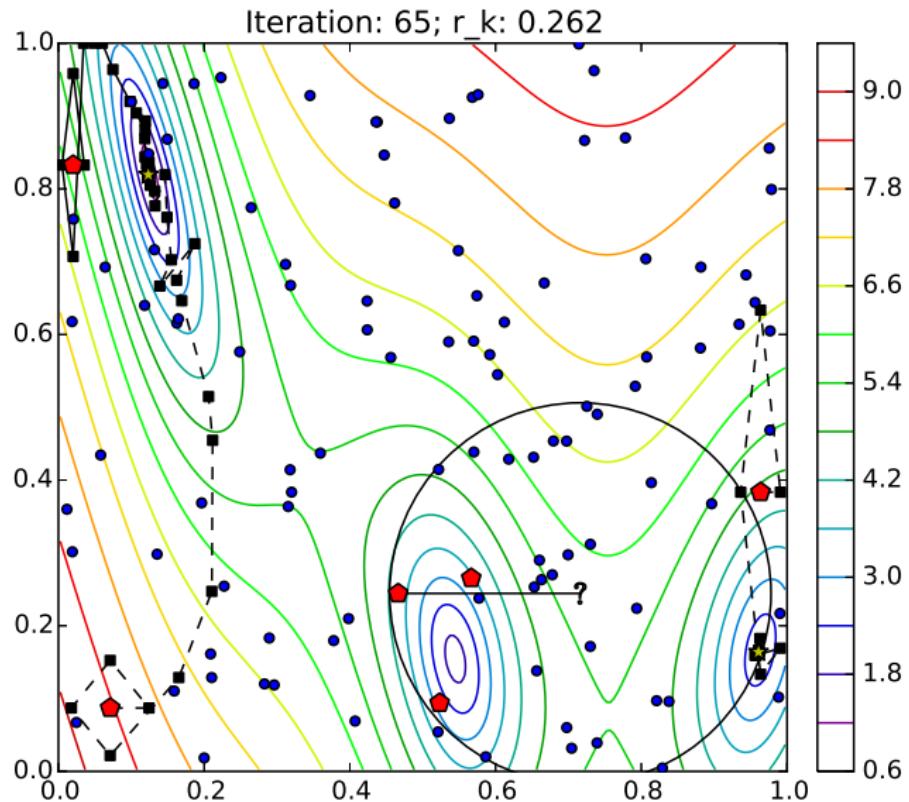
## Pausing runs



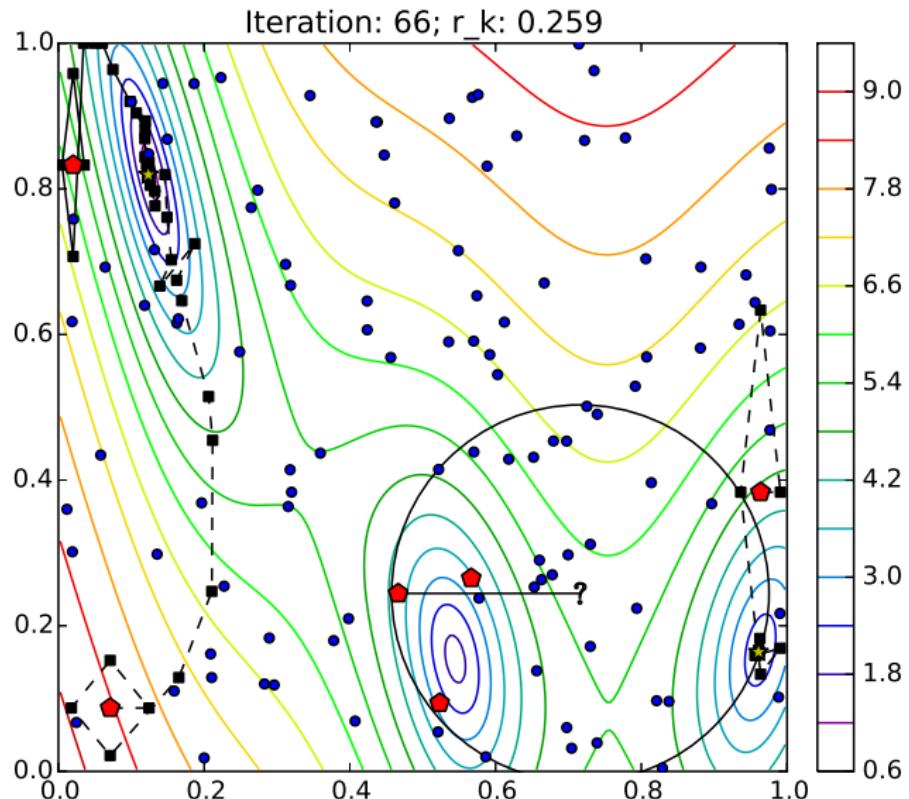
## Pausing runs



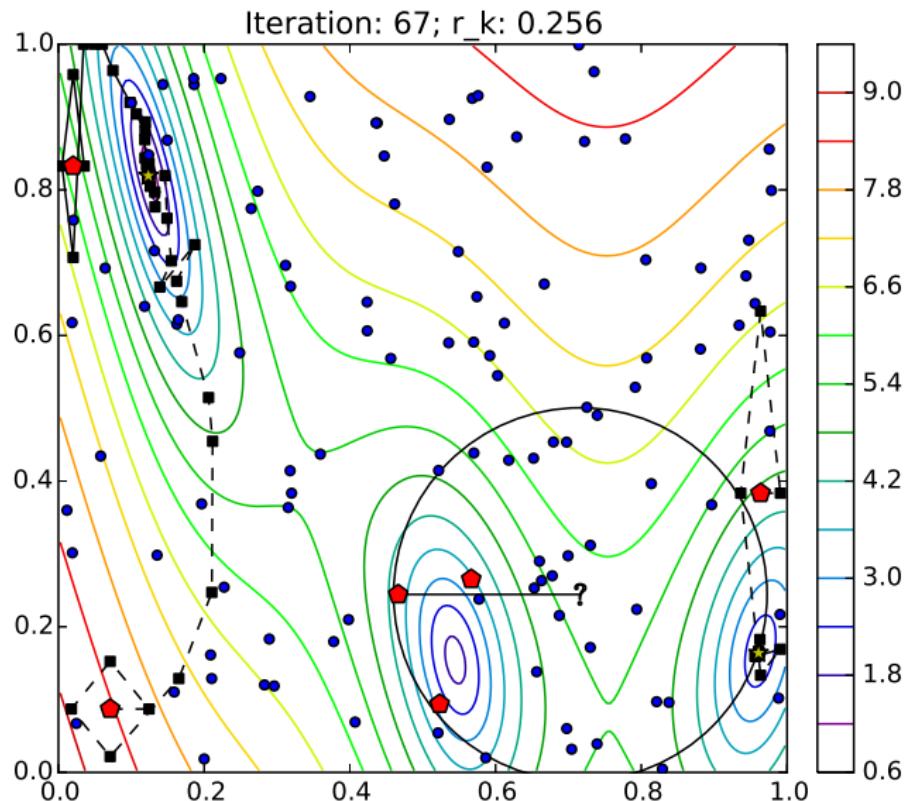
## Pausing runs



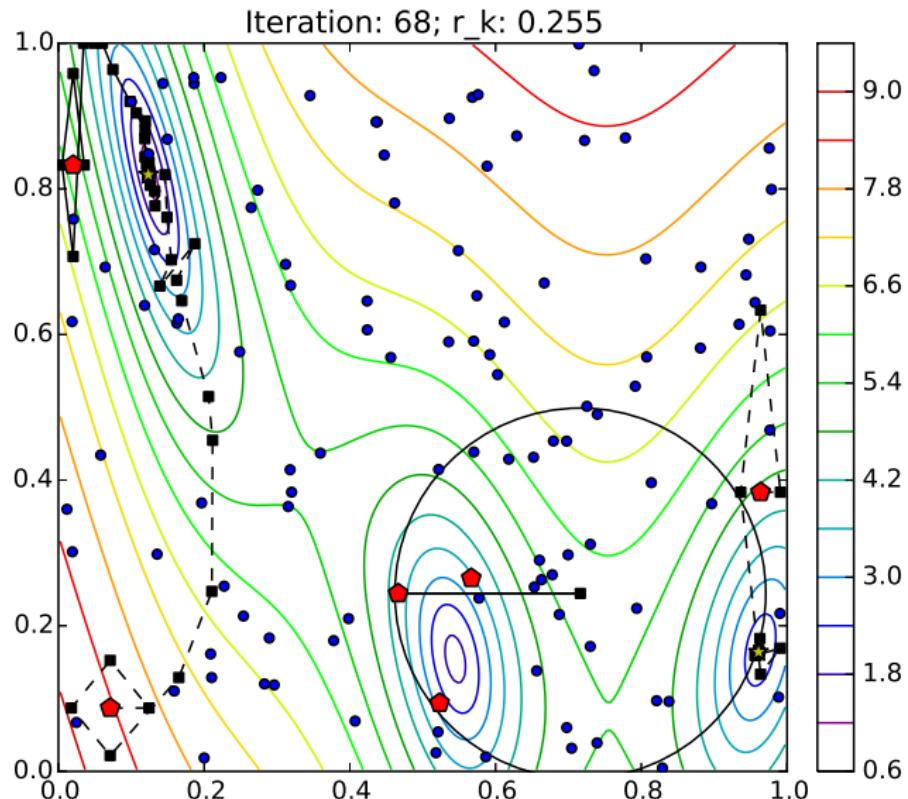
## Pausing runs



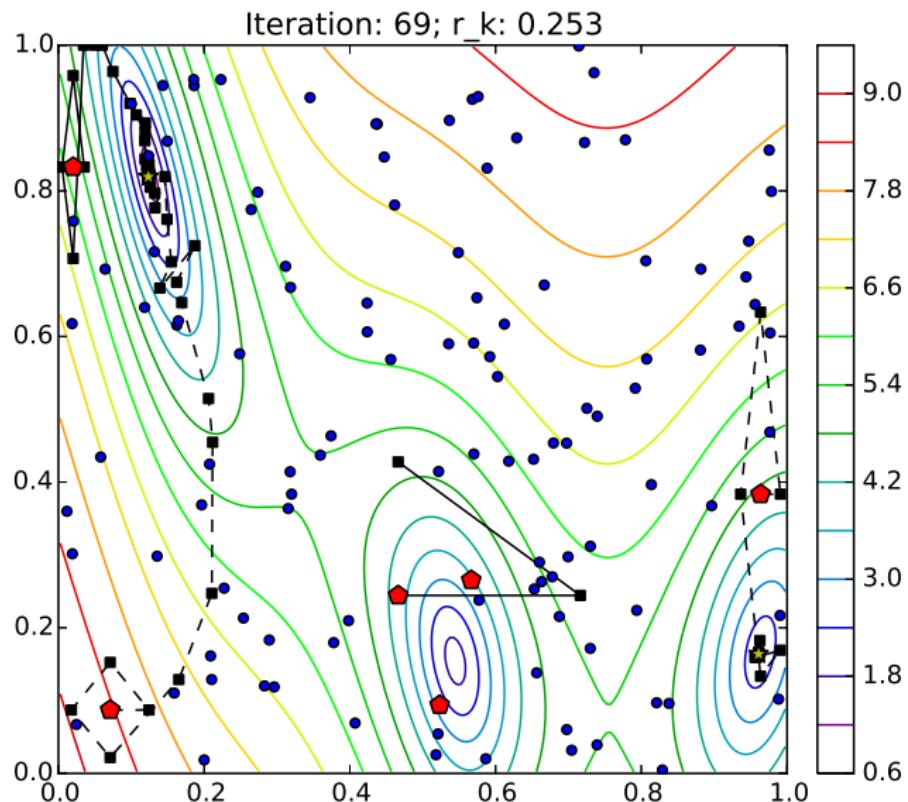
## Pausing runs



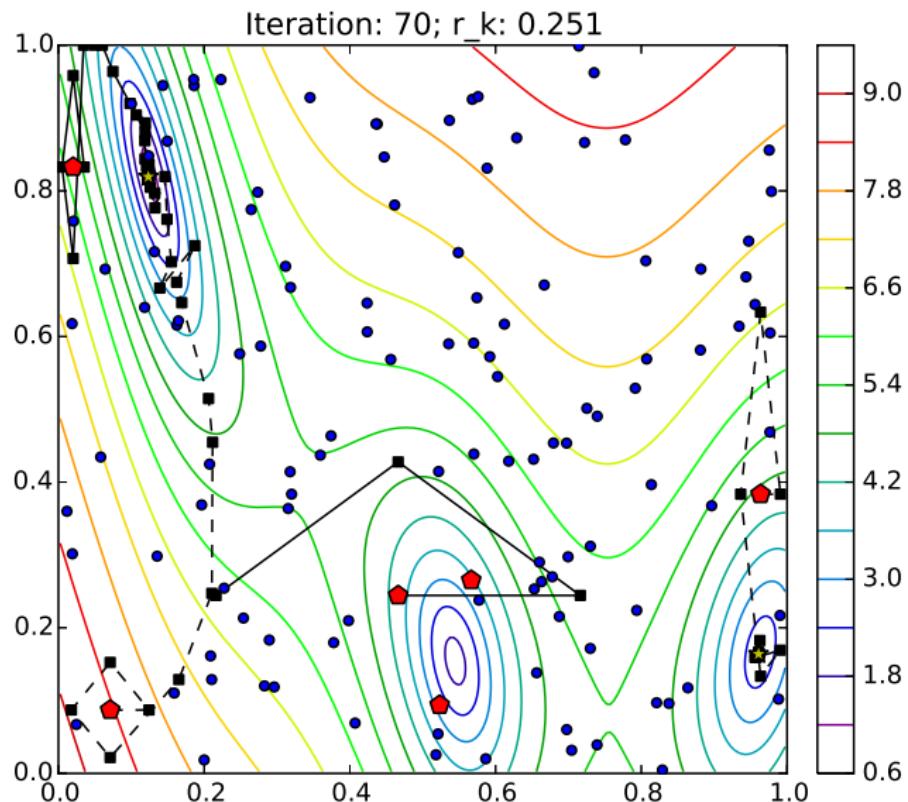
## Pausing runs



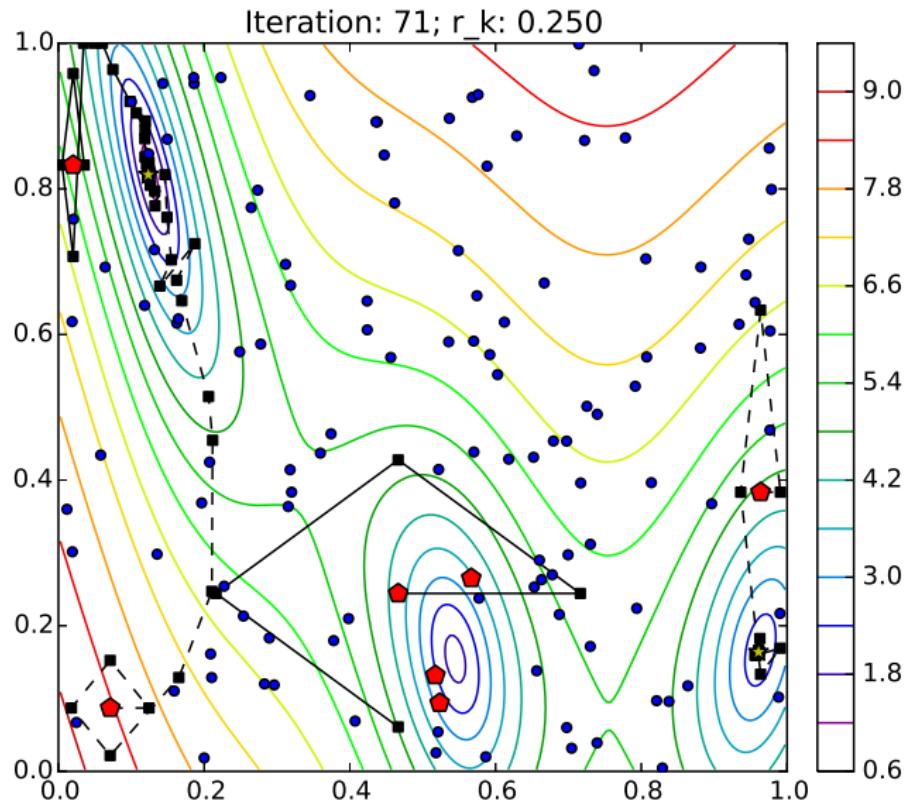
## Pausing runs



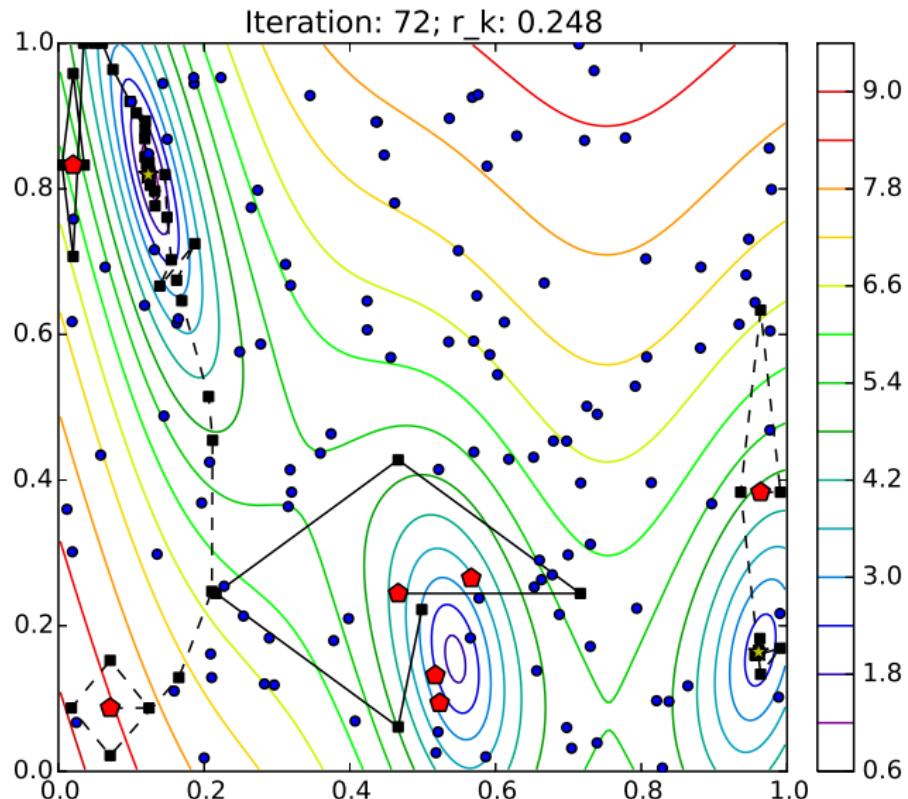
## Pausing runs



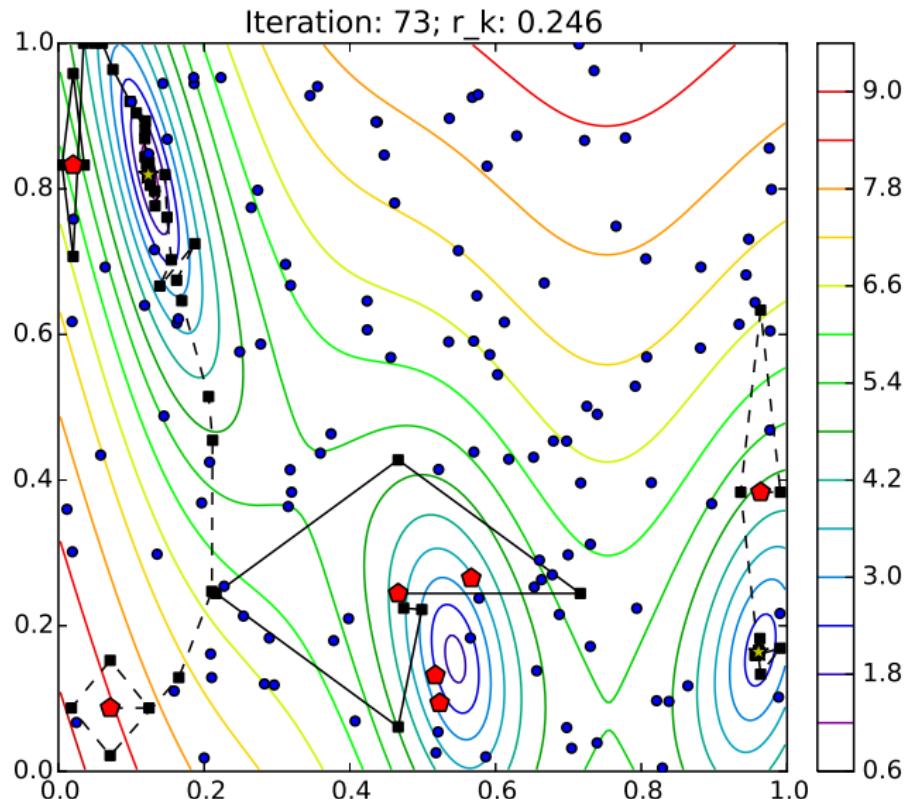
## Pausing runs



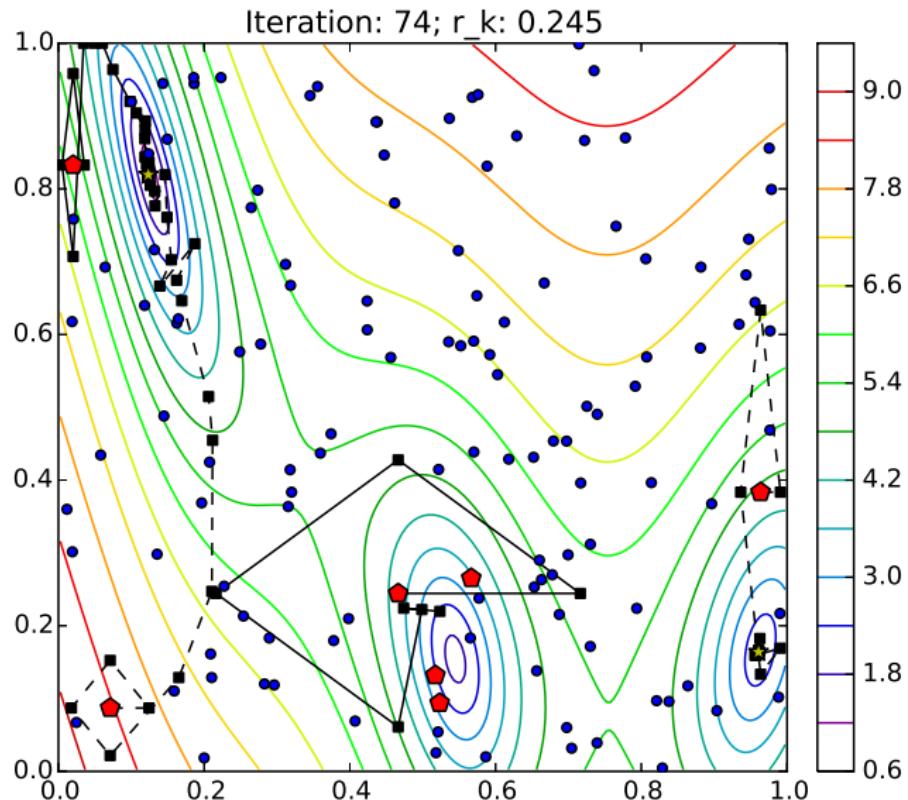
## Pausing runs



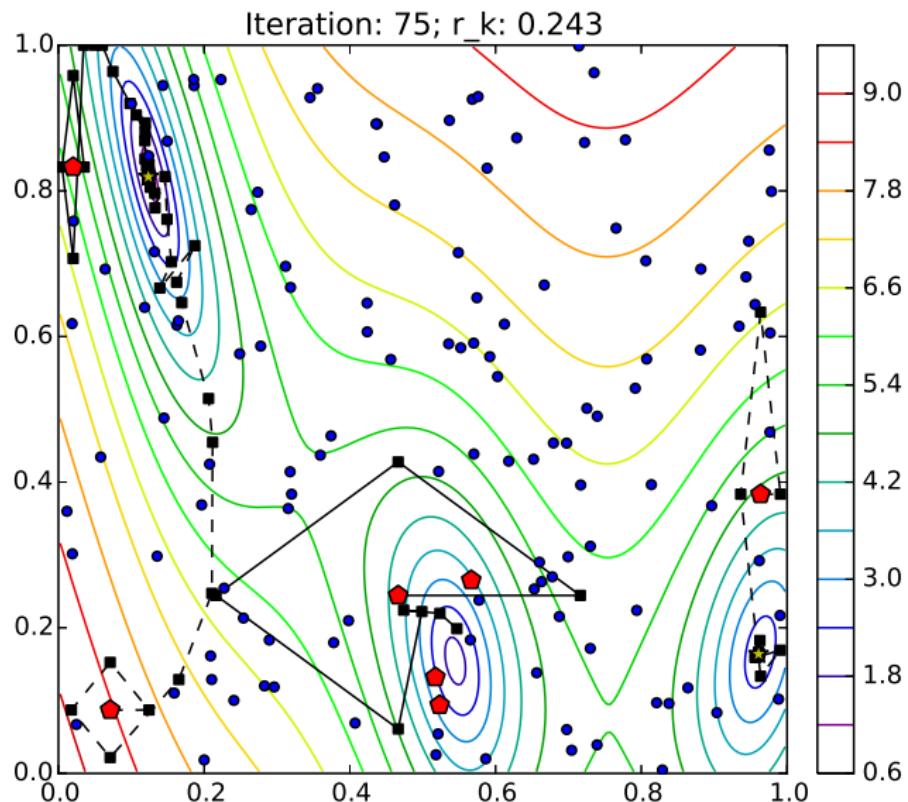
## Pausing runs



## Pausing runs



## Pausing runs



## Closing Remarks

- ▶ Concurrent function evaluations can locate multiple minima while efficiently finding the global minimum.

### Questions:

- ▶ Finding (or designing) the best local solver for our framework?
- ▶ Best way to process the queue? May involve (better) rules for pausing local runs

